uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática
2017-2018

Trabajo Fin de Grado

"Desarrollo de herramienta para generación de bases de datos de nubes de puntos anotadas para vehículos autónomos a partir de modelos sintéticos"

Pablo Muñoz Real

Tutor/es

Jorge Beltrán de la Cita

Leganés, 16 de octubre de 2018

RESUMEN

El creciente avance en computación ha llevado a un incremento del uso de algoritmos de inteligencia artificial en todos los campos de estudio. Tanto es así que ahora pueden aplicarse estos algoritmos para realizar tareas eminentemente complejas como es la conducción autónoma.

Estos algoritmos requieren de un número muy elevado de datos etiquetados durante la etapa de entrenamiento, de tal manera que el modelo resultante se comporte de forma robusta en todos los escenarios de conducción posibles. La obtención de dichos datos se realiza, habitualmente, mediante el etiquetado manual de la información capturada por sensores embarcados en vehículos, siendo ésta una tarea lenta, tediosa y de grandes costes.

La aplicación presentada en este trabajo surge de la necesidad de obtener bases de datos etiquetadas a mayor velocidad. En concreto, está enfocada en sensores láser 3D, dado el tiempo requerido por el proceso de anotación del tipo de información que capturan. Para ello se ha desarrollado una herramienta capaz de generar nubes de puntos semejantes a las capturadas por sensores LiDAR reales a partir de modelos en 3D sintéticos, permitiendo así automatizar la creación de grandes conjuntos de datos etiquetados que pueden ser utilizados para entrenar algoritmos de inteligencia artificial, sin tener que recopilar tantos datos en el terreno.

Palabras clave

Inteligencia Artificial, LiDAR, Bases de datos, Simulación, Seguridad Vial

ABSTRACT

The recent increase in computation advance has led to an extended use of artificial intelligence algorithms in every field. The increase is such that it can now be applied into complex labors such as autonomous driving.

These algorithms require a very high amount of labeled data during the training phase in order for the resulting model to behave robustly in every driving scenario. The labeled data acquisition is a slow, tiresome and highly expensive task because it is usually done manually with the information of the sensors in the vehicles.

The program presented in this thesis grew out from the need of acquiring labeled data faster and cheaper. It is focused on 3D laser sensors as its data labeling process is very time-consuming. In order to achieve that, a tool was developed to generate point clouds from synthetic 3D models, similar to those acquired by real LiDAR sensors. This allows obtaining automatically a great amount of labeled data that can be used to train artificial intelligence algorithms without having to gather data on the field.

Keywords

Artificial Intelligence, LiDAR, Datasets, Simulation, Road Safety

DEDICATORIA

A mi familia y amigos por aguantarme, apoyarme y motivarme durante estos años de estudio. A mis profesores y tutor por haberme enseñado y haberme dado las guías necesarias para aprender.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación del trabajo.

Actualmente los medios de transporte son un elemento clave para la sociedad, ya sean como medio de trabajo o para el disfrute. En 2015 había 1 200 millones de vehículos de motor en circulación [1] y se estima que para 2035 esa cifre aumente hasta 2 000 millones [2]. Por lo tanto, las congestiones y los problemas ambientales que derivan de un número tan elevado de vehículos van a incrementar en el futuro.

Una de las problemáticas principales de un número tan elevado de vehículos son los accidentes de tráfico. Actualmente se estima que mueren anualmente en tránsito alrededor de 3500 personas, número que parece estar estabilizándose [3]. Esto es gracias a las medidas de seguridad vial y al aumento de la seguridad de los vehículos.

Estos datos han mejorado, en parte, debido a la incorporación de sistemas autónomos en los vehículos. Hasta ahora las mejoras venían dadas por mejores frenos, estructuras más seguras, pero no era suficiente para frenar el aumento de víctimas. Pero con la llegada de avisadores de salida de carril, frenos automáticos o esquivas automáticas las cifras consiguieron disminuir.

El reducir el factor humano ha sido clave para reducir el número de accidentes y ésta es una de las grandes razones de peso para la incorporación del vehículo autónomo. Un informe de Morgan & Stanley de 2014 [4] habla de que para 2026 se podría conseguir un coche cien por cien autónomo y marca unos pasos a seguir, (véase la figura 1).

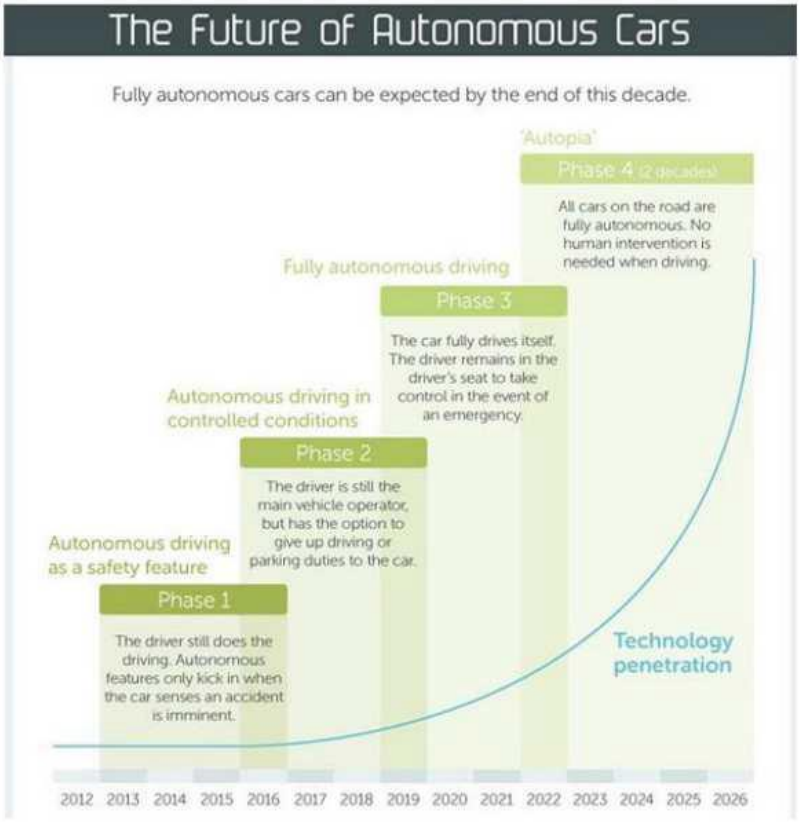

Figura 1: El futuro del coche autónomo. Artículo de Morgan & Stanley [4].

A nivel social los atascos hacen perder al ciudadano muchas horas al año, concretamente, un habitante de Ciudad de México pierde 227 horas al año en atascos [5]. Esta pérdida no es únicamente temporal puesto que, si ese tiempo se empleara en ser productivo, se estima una pérdida de 1 250 millones de dólares [6] para el caso de México.

Con vehículos autónomos no sólo se disminuiría el tiempo al volante si no también el consumo de combustible, que podría reducirse en hasta un 40% [7], lo que supondría un ahorro económico bastante importante.

El desarrollo de sistemas que mejoren la eficiencia de los automóviles es una prioridad tal que cualquier adelanto que suponga un avance en este campo actualmente adquiere una relevancia inusitada. Este proyecto pretende contribuir a este desarrollo cuyo fin es beneficiar a la sociedad mediante las ventajas del coche autónomo.

1.2. Descripción del proyecto y objetivos.

Este proyecto consiste en el desarrollo de una aplicación que a partir de un modelo 3D sintético (creado por ordenador) se pueda simular y crear una base de datos de cómo detectaría dicho modelo por un sensor láser.

Para conseguir esto se ha utilizado el lenguaje de programación C++ soportado por las librerías de PCL y Qt. La librería PCL da todos los recursos necesarios para la visualización y las operaciones a realizar con los modelos 3D sintéticos, mientras que la librería Qt ofrece todo el entorno gráfico y de interfaz de usuario.

La principal ventaja de este software es que es completamente portable porque, a pesar de haber sido programado en Linux, se puede generar un ejecutable para Windows. Aunque en principio es un programa ligero, si se tienen abiertos muchos modelos o modelos muy grandes puede llegar a ocupar mucha memoria y al iniciar la simulación precisar mucho procesamiento.

Estos son los objetivos principales:

1. Conseguir una correcta simulación de un sensor LiDAR.
2. Generar, mediante simulaciones, nubes de puntos desde distintos puntos de vista en distintas condiciones de ruido.
3. Soportar los formatos de archivos más comunes actualmente utilizados para modelos 3D.
4. . Desarrollar un software que facilite y agilice el proceso de creación de bases de datos de nubes de puntos anotadas, automatizando en gran medida el proceso.
5. Crear una base de datos de nubes de puntos a partir de modelos sintéticos.

1.3. Estructura del documento.

Este documento empieza por analizar del estado de la cuestión (apartado 2), haciendo un repaso por los sensores involucrados en el coche autónomo y cómo hoy en día interactúan con los sistemas de ayuda al conductor, y describiendo los algoritmos utilizados en detección de obstáculos y las bases de datos en las que se apoyan.

En el punto 3 se realiza la descripción de los materiales y recursos que se han utilizado para la consecución de este proyecto.

En el cuarto apartado se representa detalladamente el funcionamiento del programa, describiéndose los procesos internos y se especificando cómo utilizar el programa, explicando el significado de los parámetros necesarios.

En el capítulo 5, se detalla el proceso de simulación al completo, explicando pormenorizadamente todos los aspectos de la simulación.

En el apartado 6 se analizan los resultados obtenidos para todas las pruebas realizadas. Posteriormente se habla del presupuesto necesario (punto 8) y del marco regulador (punto 9).

Las conclusiones de este proyecto se presentan en el apartado 9 y se analiza el cumplimiento de los objetivos previamente planteados en el apartado 1.2.

Por último, se habla de los trabajos futuros (apartado 10) y cómo se podría mejorar.

2. ESTADO DE LA CUESTIÓN

Se hace preciso presentar en este documento las diversas tecnologías que se utilizan en el desarrollo del coche autónomo para así aportar un contexto de la situación actual y justificar la relación de éstas con este proyecto.

Se comenzará exponiendo los sistemas de percepción de los vehículos y se acabará con la descripción de cómo se procesan y almacenan esos datos para su interpretación durante situaciones que acontecen durante los trayectos.

2.1. Percepción por computador.

La percepción se define como "comprender o conocer algo" [8] y es precisamente lo que se necesita que haga el ordenador para permitir navegar por un entorno desconocido. La percepción puede ser tanto visual como espacial, y, para conseguirla se necesitan distintos tipos de sensores que proporcionen informaciones diferentes.

Para ello se van a analizar diversas formas de obtención de la información que un computador precisa para percibir el entorno.

2.1.1. Cámaras

Mediante una cámara conectada a un computador se capturan imágenes que posteriormente se procesan, analizando la información. Lo más actual es utilizar la cámara del teléfono móvil para obtener información de la carretera y advertirnos de posibles riesgos.

IOnRoad Augmented Driving es una aplicación que se puede encontrar para dispositivos Android en la tienda Google Play y le indica al conductor la velocidad a la que circula y a cuánta distancia se encuentra el vehículo que le precede. Además, si se encontrara el conductor muy cerca siguiente vehículo o detectara que se está saliendo del carril, la aplicación emitiría tanto una señal de alerta en pantalla como otra acústica para advertir del peligro.

Pero si bien es verdad que los móviles con cámara son algo muy frecuente, la completa identificación de un peligro y su esquiva sigue dependiendo de la pericia del conductor. Por eso muchos coches ya incorporan cámaras que asisten al conductor ante determinadas acciones. La detección y posterior análisis por la computadora no siempre provocan una respuesta automatizada, sino que ofrecen información al conductor. Ejemplo de ello son las cámaras de marcha atrás o incluso laterales, como las ofrecidas por BMW para mejorar la visibilidad en cruces [9]. Otro ejemplo son los sistemas de reconocimiento de señales que ofrecen algunas marcas (Opel, Ford...) que son capaces de informar al conductor si supera el límite de velocidad.

Tanto Citroën como Toyota las utilizan las cámaras para advertir del cambio de carril. El sistema de Toyota [10] consiste en una cámara en el espejo retrovisor central que detecta los carriles, tanto blancos como amarillos. Si el sistema interpreta que el vehículo se está desviando emite un pitido y una de las luces de cambio de carril del panel de información

empezará a parpadear. Si el coche también está equipado con el asistente de giro automáticamente, este entrará en acción y recolocará de nuevo el vehículo en el centro del carril. Más adelante se detallarán otros sistemas que existen de percibir el cambio de carril.

2.1.2. Cámaras estéreo

Se trata de una variación del sistema anterior al que se le añade una segunda cámara. Su funcionamiento también se basa en la toma de imágenes, pero, al estar las cámaras separadas a una distancia fija, estas pueden percibir, además de la información individual en 2 dimensiones de cada fotografía, la profundidad a la que se encuentran los objetos. Por eso muchos fabricantes de automóviles las utilizan para la detección de objetos.

Land Rover incorpora una cámara estéreo para la asistencia a la frenada fabricada por Bosch [11]. Si el sistema detecta un obstáculo accionará el freno automáticamente de no haber sido accionado antes por el conductor.

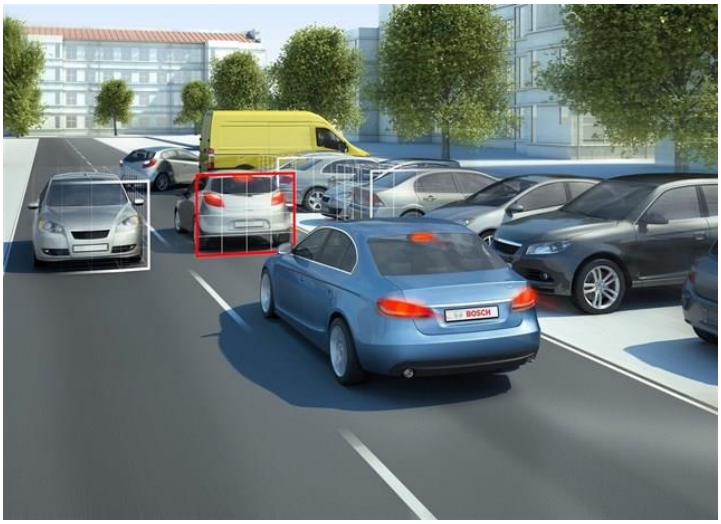

Figura 2 Simulación de situación de frenado. [11]

2.1.3. Radares

Los radares emiten ondas electromagnéticas que al reflejarse con un objeto dan información sobre la distancia a la que está dicho objeto. Con un segundo reflejo se puede saber también la velocidad a la que éste se aproxima. Aunque es un sistema que no permite conocer plenamente el entorno, para la ejecución de tareas simples es suficiente y reduce el coste computacional, utilizándose en muchos sistemas de aparcado automático o frenado de emergencia. También encontramos esta tecnología en algunos sistemas de crucero adaptativo, un sistema que mantiene la velocidad constante, pero adaptándola para mantener la distancia de seguridad con el vehículo de delante.

Para los sistemas de aviso de ángulos muertos también se suelen usar estos sensores puesto que es una información rápida y directa, solo se necesita saber algo ocupa ese espacio no visible por el conductor, no las características del objeto.

2.1.4. LiDAR

Viene nombrado por su acrónimo en inglés de detección de luz y distancia (*Light Detection And Ranging*). Básicamente lo que hace es emitir muchos haces y medir el tiempo que tardan en llegar a su objetivo. De esta manera se obtiene información muy precisa del terreno que se haya analizado, utilizándose como sensor clave para la navegación. Tanto es así, que empresas como Ford y General Motors han apostado por desarrollar un LiDAR más barato y eficiente, mientras otras como Tesla siguen negándose a utilizarlo apostando por las cámaras y los radares [12]. El desarrollo de este trabajo se ha realizado, precisamente, enfocado en estos sensores.

2.2. Algoritmos de clasificación de obstáculos

Para tener en cuenta todas las condiciones que se suceden cuando un vehículo circula del punto A al punto B es necesario un conocimiento del entorno muy extenso, siendo clave la detección de obstáculos, para ello hay que clasificar toda esa información. Se van a analizar los algoritmos más prometedores, con especial atención a las CNN (*Convolutional Neural Network* o Redes neuronales convolucionales) puesto que es un concepto recurrente en algoritmos más avanzados.

2.2.1. PCA

Principle Component Analisis o Análisis de Componentes Principales en castellano es un algoritmo con el que se reduce las dimensiones de los datos observados. Para ello realiza una transformación lineal ortogonal a un nuevo sistema de coordenadas. La primera componente es la dirección en la que se produce la variación máxima y la segunda es la segunda mayor variación. [13]

Esta técnica es efectiva para el reconocimiento de patrones y objetos, pero la separación y clasificación son a menudo complicadas y producen errores.

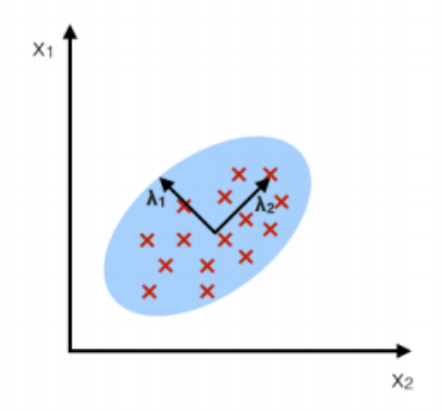

Figura 3 Colección con dos componentes principales λ [13]

2.2.2. LDA

Linear Discriminant Analysis o Análisis Discriminante Lineal es un método basado en la apariencia que se usa para la reducción de dimensiones o clasificación. Cada clase se representa por su media y una covarianza. Mientras se minimiza la varianza común se

maximiza la varianza de cada clase. Así al proyectar las clases se obtiene la separación. Una proyección buena es aquella que separa bien las clases. A continuación, se puede ver un ejemplo. El óvalo coloreado representa cada clase. [13]

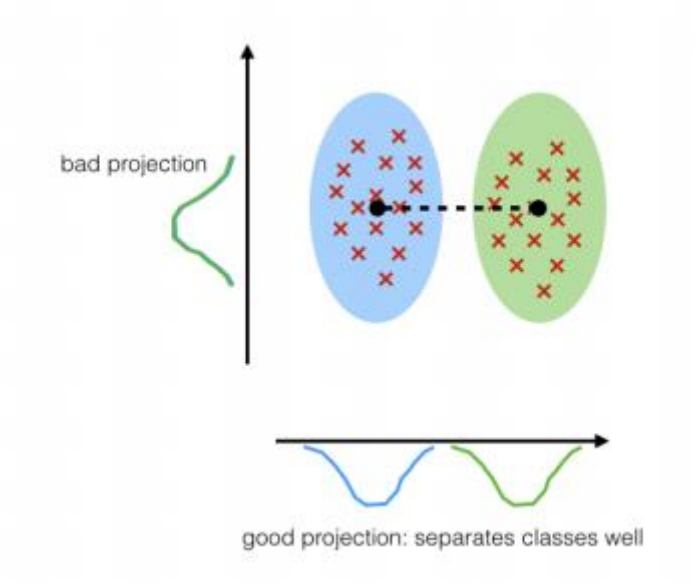

Figura 4 Buena y mala separación de dos clases [13]

2.2.3. CNN

Las Redes Neuronales Convolucionales están inspiradas en la estructura del cerebro humano que constantemente predice y clasifica situaciones y objetos. De la misma manera que un niño pequeño aprende a reconocer su entorno, una CNN necesita muchos datos para ser capaz de identificar un patrón o una imagen. Para ello se le dan miles de ejemplos ya clasificados. Este proceso se le denomina entrenar la red neuronal.

Este tipo de redes son muy útiles para procesar directamente imágenes, texto o sonido puesto que permiten combinar información local y global, es decir el contexto con zonas de detalle. Sirven para captar patrones con facilidad y es por eso que en imágenes se utilizan para detectar rostros, objetos o situaciones.

Las redes neuronales convolucionales constan de varias capas, que dependiendo de la aplicación pueden variar de número y de tipos de conexiones. Las capas se organizan en 3 dimensiones, todas las neuronas de una capa no tienen por qué estar conectadas a todas las neuronas de la siguiente capa y al final se reduce a un vector de probabilidad. [14]

Las capas pueden ser de 3 tipos:

- Capa de entrada: recibe los datos originales.
- Capa de salida: proporcionan la respuesta.
- Capa oculta: Las capas intermedias donde se procesa la información.

Según el número de capas de que se encuentren en la red neuronal podemos hablar de *deep learning* o aprendizaje profundo. Cuantas más capas ocultas existan, más profundo y más complejo es el proceso de entrenamiento. [15]

Para la conducción autónoma este proceso es muy importante, porque no solo se puede aplicar a la detección de obstáculos, sino que puede aplicarse a toda la conducción con un número suficiente de datos de entrenamiento.

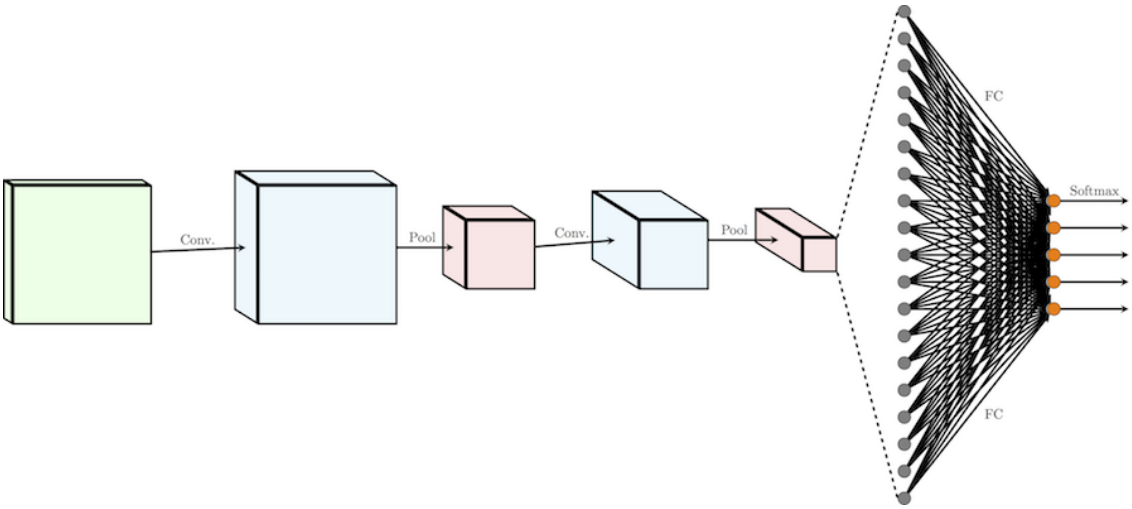

Figura 5 Esquema de red neuronal convolucional. [16]

2.2.2. Detección de obstáculos utilizando imágenes

Un ejemplo de cómo utilizar CNN para reconocimiento de obstáculos lo tenemos en el algoritmo desarrollado por S. Ren [17] que se aprovecha de analizar las imágenes basándose en regiones y compartiendo capas de la CNN. Precisamente compartir capas es lo que diferencia este algoritmo de otros y es lo que ha conseguido que el coste computacional que supone este tipo de redes neuronales se reduzca considerablemente. Con este algoritmo se consigue un procesamiento casi en tiempo real llegando a funcionar a 5-17 fps. Este método también mejora la calidad de las detecciones frente a una CNN clásica.

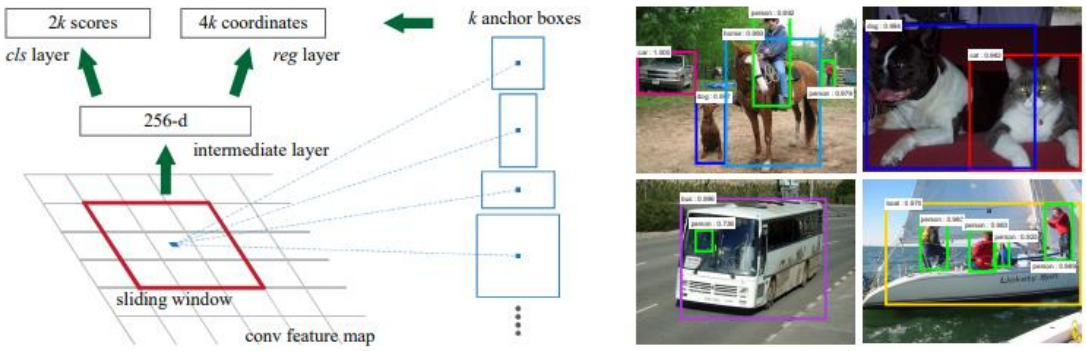

Figura 6 A la izquierda la región propuesta de estudio y a la derecha ejemplos de detección [17]

2.2.3. Detección de obstáculos utilizando LiDAR

J. Beltrán y su equipo [18] han desarrollado BirdNet, un algoritmo que basándose en datos de LiDAR genera nubes de puntos desde una vista aérea y posteriormente entrena una CNN para detectar y clasificar obstáculos. Este algoritmo se ha probado con la base de datos de KITTI, de la que se hablará en el apartado de bases de datos, obteniéndose unos resultados muy prometedores.

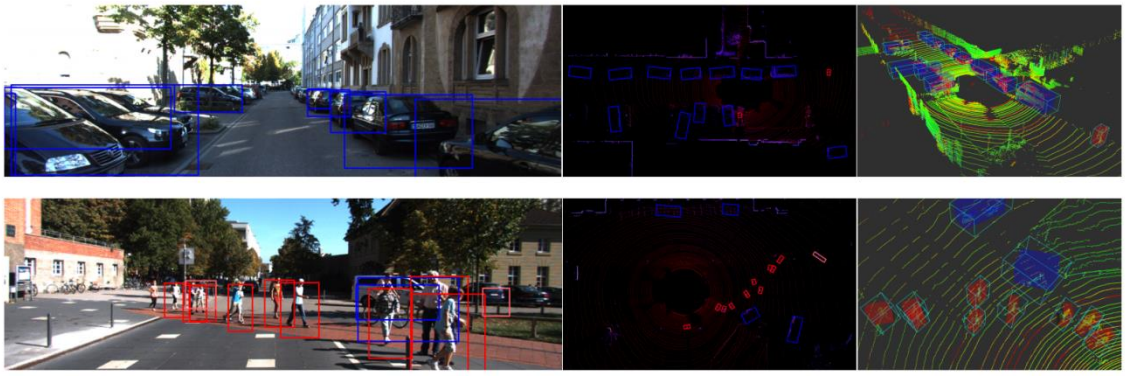

Figura 7 De izquierda a derecha: Detecciones en la imagen, vista de pájaro y nube de puntos. [18]

2.2.4. Detección de obstáculos utilizando LiDAR e imágenes

Frustum PointNets [19] es un proyecto por el cual mezclando datos de LiDAR y una cámara obtienen resultados de vanguardia. Primero con la cámara se genera un área de interés que posteriormente con los datos 3D obtenidos por el LiDAR se puede limitar la región en 3D en la que se sitúa el objeto.

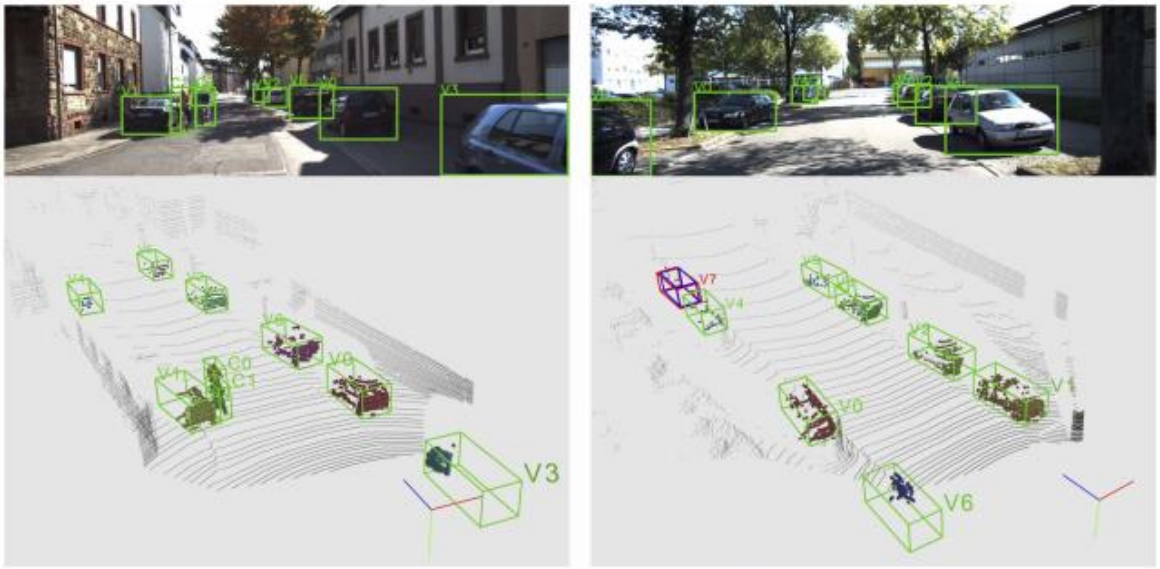

Figura 8 Ejemplo de cómo se delimitan los obstáculos detectados. [19]

2.3. Bases de datos

Para entrenar una inteligencia artificial se necesita una gran cantidad de información etiquetada y por ello existen numerosas bases de datos que se han ido desarrollando durante años específicamente para ayudar con la conducción autónoma. En este apartado se hablará de los distintos tipos de bases de datos que existen y cómo son útiles para el desarrollo del coche autónomo.

2.3.1. Visión estéreo/3D

En la universidad de Middlebury, Scharstein y sus colaboradores han generado y mantenido una base de datos con la que, utilizando un sistema de iluminación estructurado, se consiguen generar varias bases de datos de imágenes de escenas de interior a partir de correspondencia de subpixels en 2D. También sirve para el calibrado de la información de cámaras y proyectores. La última revisión de este conjunto es de 2014 donde se consigue una evaluación más precisa y una mayor rectificación de los conjuntos anteriormente generados. [20]

Figura 9 Imagen de una moto generada de la base de datos de Scharstein [20]

El trabajo generado en Middlebury por Seitz y su equipo es una base de datos estéreo multivista que proporciona datos de gran calidad con los que poder entrenar y evaluar distintos algoritmos de reconstrucción multicámara. También sirve de marco de referencia para analizar algoritmos novedosos dentro de esa materia. [21]

Figura 10 Imagen de un templo generada por Seitz. [21]

Citiscapes es un dataset de escenas en la calle recogido en 50 ciudades diferentes con cerca de 5000 anotaciones. Con esta base de datos se pretende dar información para redes neuronales convolucionales profundas y probar el rendimiento de distintos algoritmos de visión por computador. [22]

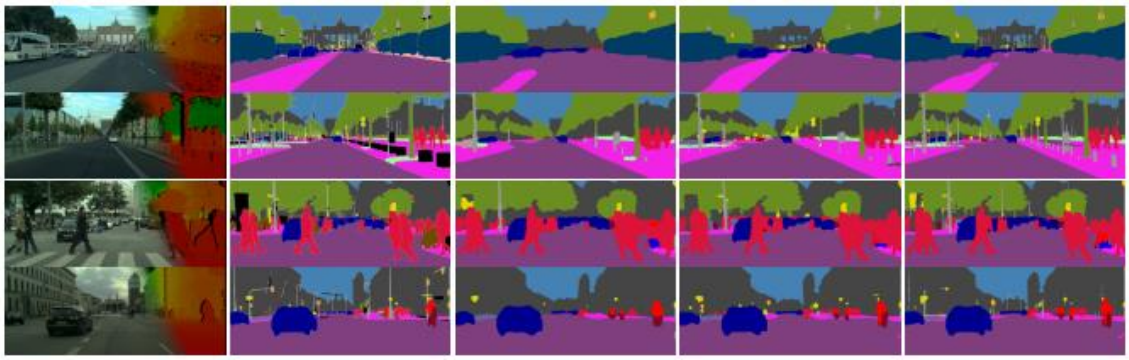

Figura 11 Ejemplo de la base de datos de Cityscapes. [22]

2.3.2. ImageNet y COCO

ImageNet es una base de datos de imágenes etiquetadas por palabras, en su mayoría sustantivos. Dispone de más de 14 millones de imágenes clasificadas en 22 mil categorías distintas. El objetivo es ayudar a los investigadores académicos de tratamiento de imágenes. Por ejemplo, estas imágenes pueden servir para el clasificado de redes neuronales convolucionales que detecten obstáculos. [23]

Figura 12 Ejemplo de ImageNet [23]

En la misma línea que ImageNet se encuentra COCO, pero esta vez la imagen aparecen los objetos etiquetados dentro de un contexto, no solo la imagen recortada como en

ImageNet. Tiene cerca de 1.5 millones de objetos en 330 mil imágenes clasificados en 91 categorías. [24]

Figura 13 Ejemplo de etiquetado de COCO [24]

2.3.3. KITTI

Es una plataforma a través la cual se han conseguido datos sobre muchos sensores y cámaras involucrados en el desarrollo del coche autónomo. Las bases de datos se han generado montando sobre un vehículo cámaras estéreo, tanto a color como en blanco y negro, un LiDAR y un sistema de localización GPS y conduciendo por la ciudad de Karlsruhe, zonas rurales y autovías. Los datos se obtienen a una velocidad de 10 Hz y se guardan en ordenador equipado con Ubuntu y una base de datos a tiempo real. [25]

De esta manera se han conseguido diversas bases de datos relacionadas con el coche autónomo. Las imágenes estéreo se han etiquetado manualmente para poder identificar peatones, coches, furgonetas, camiones, ciclistas y otros obstáculos.

Figura 14 Ejemplo de etiquetado de la base de datos de Scene Flow de KITTI [25]

3. MATERIALES Y RECURSOS

En este apartado se va a hablar de los distintos recursos utilizados para realizar el proyecto, tanto software como hardware.

3.1. Software

Se han utilizado una serie de aplicaciones informáticas que se detallan a continuación.

3.1.1. Qt creator

Figura 15 Logo Qt [26]

Qt [26] es una plataforma de programación que tiene dos modalidades, la comercial y la Open Source. Tiene soporte para diversos sistemas operativos como Windows, Linux y MacOS.

La versión Open Source incluye acceso a todas las librerías y APIs de interfaz de usuario y componentes de desarrollo. También incluye ciertas herramientas y características que no están limitadas. Como se distribuye bajo licencia LGPL [27] hay que ser consciente de sus términos y condiciones legales.

La versión comercial incluye acceso a la versión Open Source, pero se añaden el resto de herramientas y características, además de herramientas y soluciones para sistemas embebidos, soporte oficial de Qt y una estrategia cercana con Qt como compañía. Se incluye una versión de prueba de entre 10 y 30 días para probar sus funciones. A partir de, y dependiendo de los servicios que se contratan, cuesta 459 dólares al mes.

Para este proyecto solo se ha necesitado la versión Open Source puesto que únicamente se necesitaba acceso a la aplicación de programación y a sus librerías de interfaz de usuario que se distribuye bajo licencia LGPL.

3.1.2. Librería PCL

Figura 16 Logo PCL [28]

PCL [28] es un proyecto de procesamiento de imágenes en 2D y 3D. Se distribuye bajo licencia BSD [29] por la cual se puede utilizar gratuitamente tanto para fines comerciales como de investigación. Es multiplataforma, lo que quiere decir que funciona en Linux, MacOS, Windows, iOS y Android. Se subdivide en varias librerías para poder ser distribuida en plataformas con problemas de espacio y/o computación.

PCL se desarrolla por diversas organizaciones distribuidas por todo el mundo, como son la Universidad Carlos III de Madrid, Intel, Nvidia, Toyota o Velodyne entre otros; financiándose de sus socios, siendo dos de ellos Google y Toyota, o mediante donativos de cualquier persona interesada en apoyar el proyecto.

En esta librería se basa todo el grueso del proyecto. Es la encargada de cargar los modelos y realizar las simulaciones.

3.1.3. C++ Standard Library

La librería estándar de C++ ha sido extensivamente utilizada puesto que al estar programado el proyecto entero en C++ era imprescindible su uso. Concretamente se ha hecho uso principalmente de "vector", para crear vectores de las clases utilizadas, y "cout" para los mensajes de error por consola de comandos entre otros usos.

3.2. Hardware

En esta sección se detallará el hardware utilizado, cambiado a mitad de proyecto. Se empezó con un portátil HP Split X2 13-m210eg que luego fue sustituido por un ordenador de sobremesa personalizado.

El portátil tiene las especificaciones siguientes:

- Ubuntu 16.04.5
- Disco duro principal 64 GB (130MB/s)
- Memoria RAM DDR3L 8 GB
- Pantalla de 13 pulgadas (1366x768 pixel)
- CPU: Intel Core i5-4210Y (doble core 1.6GHz)
- GPU: Intel HD Graphics 4200

Como las características del portátil, aunque mejoradas con respecto del fabricado en serie con un aumento de RAM, son bastante bajas para el nivel de procesamiento que exigía la aplicación e imposibilitaba cargar modelos de tamaño considerable en un tiempo razonable. Cuando falló el disco duro del portátil se optó por utilizar un PC de sobremesa con mejores características.

El PC de sobremesa tiene las siguientes especificaciones:

- Ubuntu 16.04.5
- Disco duro principal 160 GB (3.0 GB/s)
- Memoria RAM DDR3 12 GB
- Pantalla de 20 pulgadas (1366x768 pixel)
- CPU: Intel Core i7 930 (4 core 2.8 GHz)
- GPU: AMD Radeon HD 5800 Series
- Ventiladores de 12 pulgadas.

Con el cambio de hardware se pudo notar una notable disminución del tiempo de carga de modelos grandes debido a un disco duro y procesadores más rápidos. También los tiempos de simulación se vieron reducidos considerablemente.

4. DESCRIPCIÓN DETALLADA DE FUNCIONAMIENTO

En este apartado se presentará de manera pormenorizada la forma de trabajar con la aplicación. Para facilitar el uso de la aplicación se ha desarrollado una interfaz gráfica apoyándose en la librería de Qt. Primero se detallará el funcionamiento general para posteriormente ir explicando en detalle cada parte del programa.

4.1. Funcionamiento general

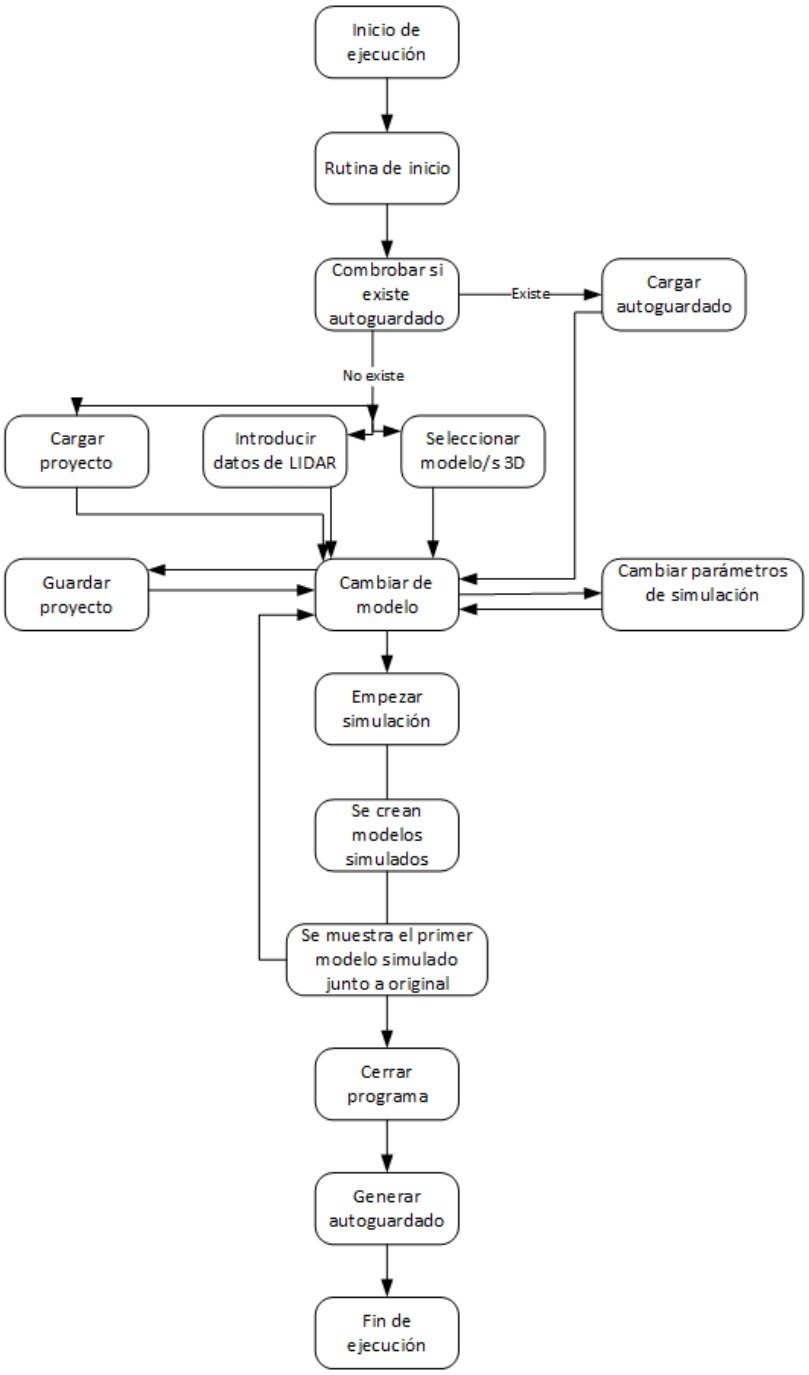

Figura 17 Funcionamiento general de la aplicación.

Al abrir el programa se ejecuta la rutina de inicio en la que se configuran los aspectos básicos de la aplicación, aspectos que serán detallados más en profundidad en el próximo apartado. En este momento inicial, siempre que haya sido usada con anterioridad, se carga el estado de la aplicación en el momento en el que fue cerrada. Esto carga tanto el sensor utilizado como las condiciones de la simulación y los modelos anteriormente abiertos.

Si no existiera un estado previo de la aplicación que cargar automáticamente se puede cargar un proyecto guardado. Los datos del proyecto se guardan en dos archivos, uno con la información de los modelos y otro con la del sensor. Para cargar un proyecto se abre un diálogo de selección de archivos que filtra los formatos de ficheros para facilitar la selección del archivo deseado.

En caso de no haber información sobre el sensor habría que introducir los datos que se vayan a analizar. Desde el menú de configuración de LiDAR se podrán establecer todos los parámetros. Si los planos del sensor están distribuidos de manera uniforme existe la posibilidad de marcar la casilla de "Planos regulares" para que los planos se distribuyan de manera automática en sus correspondientes ángulos. Es importante tener en cuenta que, si únicamente se dispone de un plano, los ángulos verticales máximos y mínimos han de coincidir con el ángulo en el que se encuentra el plano. De la misma manera, si únicamente existiera un haz de laser por plano, el "Azimut" debería ser igual a la máxima apertura horizontal, que por defecto son 360º.

Para cargar modelos existen dos posibilidades, cargarlos individualmente o agrupados en una carpeta de modelos. Para ambas opciones se abre un diálogo de selección y el usuario escoge uno o varios modelos. Se puede navegar a través de los modelos con los botones de "Previous" y "Next", a través del menú "Edit" o con el atajo rápido (Alt + flecha izquierda o derecha). No todos los modelos son soportados por la aplicación, en el caso de que el usuario intentase abrir uno no soportado un mensaje de error le advertirá de la incompatibilidad.

Para cambiar los límites espaciales de la simulación el usuario debe ir a la barra de menú y abrir en la pestaña de LiDAR, "Position Configuration". Una vez ahí se puede configurar el movimiento del sensor por el espacio y su inclinación durante las simulaciones.

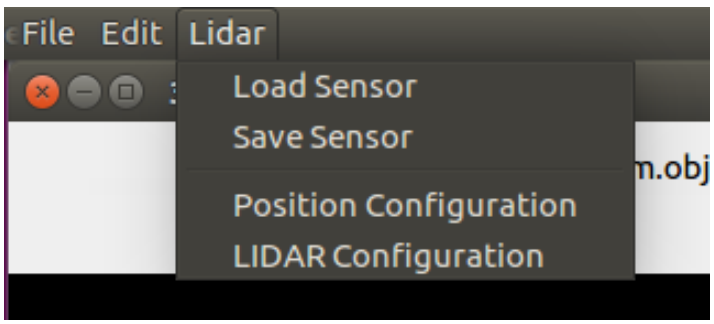

Figura 18 Menú LiDAR.

Para iniciar la simulación el usuario ha de hacer click sobre el botón de “Start Simulation”. Cuando la simulación haya terminado aparecerá un mensaje y se mostrará el primer resultado que obtenido. Entonces se podrá seguir con otros modelos o cambiando configuraciones del sensor. Se hablará con más detenimiento de la simulación en el apartado 5.

En cualquier momento se puede cerrar el programa, tanto con el botón de “Quit”, como con el atajo rápido (Ctrl + Q). Entonces se guardarán todos los parámetros del programa y los sensores bajo el autoguardado.

4.2. Rutina de inicio

Se trata de los procesos a seguir para que el programa se inicialice. El primer paso es la apertura de una nueva ventana con la interfaz gráfica. A continuación, se configuran los filtros de archivos soportados por la aplicación.

El siguiente paso es la creación de las conexiones entre las acciones de la barra de menú y las funciones generadas en el código, de tal manera que cuando se haga clic sobre una opción se ejecute correctamente la función a la que está asociada. En este apartado también se activan los atajos de teclado.

Para poder continuar es necesario ocultar el menú de “Position Configuration” puesto que ocupa el mismo lugar que el de “LiDAR Configuration” y no se pueden visualizarse en pantalla juntos. Llegado a este punto se crea también la nube donde se van a representar posteriormente los modelos, generándose 200 puntos aleatorios como representación temporal. La caja de visión de modelos es lo siguiente en mostrarse y dónde se muestran los puntos que se han generado como muestra.

A continuación, se conectan los botones de la interfaz con sus respectivas funciones. Los *spinboxes* son las cajas donde se introducen, entre otros valores, los del número de planos, los límites de los ángulos y la distancia máxima de muestreo. Para configurarlo y que se entienda bien tanto sus unidades como qué definen cada uno, a parte del título, se añaden prefijos y sufijos para indicar si es un máximo o un mínimo o saber si el ángulo es de *roll*, *pitch* o *yaw*. Si ese *spinbox* en concreto requiere de límites es en ese momento cuando se asignan. De la misma manera que se hizo con los botones, es necesario conectar el cambio en las casillas con la función que controla los parámetros en el código.

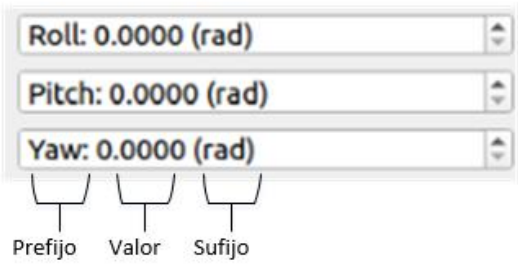

Ilustración 19 Ejemplo de *spinbox*

Como debe de haber un plano para que tenga sentido la existencia de un sensor, se genera un primer plano. También se asigna la zona de *scroll* al área donde se encuentran los *spinbox* del ángulo de los planos.

Luego se asignan los colores de las nubes de puntos, tanto de la nube principal como de la del ejemplo de simulación. Posteriormente se actualiza el visor principal para que se apliquen los colores y se cargan los autoguardados. Se deja para el final de la secuencia de inicio la carga del autoguardado para que todas las funciones del programa estén operativas a la hora de cargar los modelos y los sensores.

4.3. Interfaz gráfica

Desde el entorno gráfico se controlan todos los aspectos básicos del programa. En este apartado se explicarán todas las partes y como cambiar entre los distintos paneles de configuración.

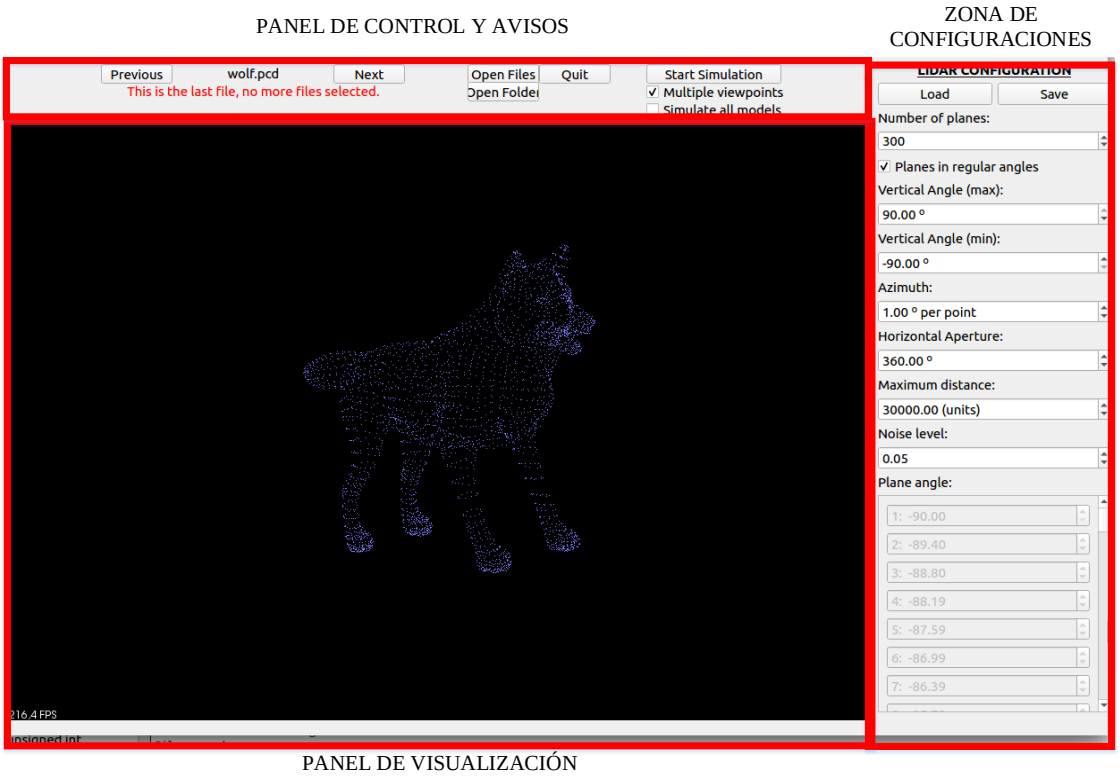

Figura 20 Partes de la interfaz gráfica.

4.3.1. Zona de visualización

Se trata del espacio en el cual se mostrarán los diferentes modelos cargados. Una vez realizada la simulación, en esta área también se visualizará el modelo simulado para así poder comparar el resultado obtenido con la muestra original y comprobar las coincidencias.

4.3.2. Barra de menú

Dicha barra se encuentra en la parte superior de la pantalla. Esta posición puede variar según qué sistema operativo utilice o cómo lo haya configurado. En Ubuntu 16.04.5 se encuentra pegada a la esquina superior izquierda de la pantalla.

La barra de menú se divide en tres submenús, “File”, “Edit y “LiDAR”. El menú de “LiDAR” se puede consultar en la Figura 18 y los otros dos en la figura 21.

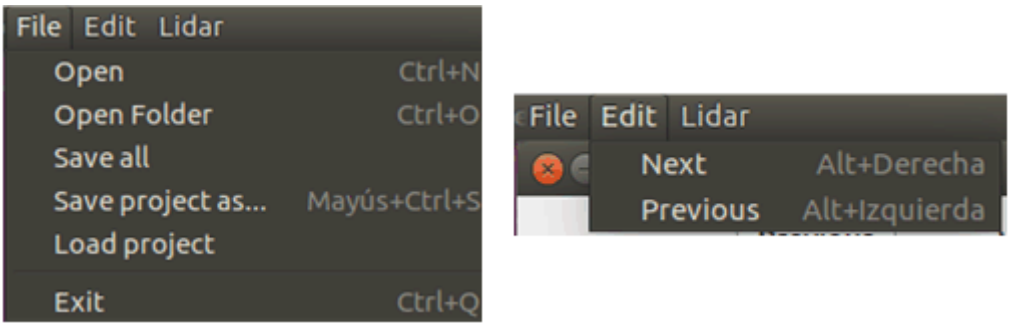

Figura 21 Menú "File" y menú "Edit"

El submenú de “File” es desde donde se pueden abrir modelos, guardar y abrir proyectos o cerrar la aplicación. También muestra los atajos de teclado para aquellas funciones que los tienen.

El submenú de “Edit” es el más simple y únicamente incluye las funciones de pasar al siguiente modelo o volver al anterior. Función que también puede hacerse con los botones del tablero o con los atajos de teclado que se muestran al desplegarlo.

Por último, en el menú de “LiDAR” se pueden cargar o guardar las configuraciones de la simulación o cambiar entre los distintos paneles de configuración.

4.3.3. Panel de control y avisos

Los botones de “Open Files” y “Open Folder” sirven para abrir archivos y carpetas respectivamente. Al pulsarlos aparece una ventana para seleccionar los archivos compatibles o una carpeta. Si se selecciona una carpeta, se abrirán sólo aquellos archivos que sean compatibles y se ignorará cualquier otro. Una vez abiertos los archivos con los botones de “Previous” y “Next” o bien con Ctrl + flecha izquierda y Ctrl + flecha derecha se puede navegar a través de los distintos modelos cargados. El nombre del archivo aparecerá entre los botones de “Previous” y “Next” y debajo de ellos los avisos que produzca el programa.

El botón de “Quit” sirve para cerrar el programa de manera normal, también se puede cerrar a través de la barra de menú como se ha explicado en el apartado de la 4.3.2 Barra de menú.

En la parte derecha del panel de control encontramos el botón “Start Simulation”, que inicia la simulación con los parámetros que se hayan establecido en los distintos paneles de configuración. La casilla seleccionable situada bajo el botón “Start Simulation” permite combinar datos. Si se encuentra desmarcada únicamente se producirá un resultado en la simulación. Sin embargo, de encontrarse seleccionada, se realizarán todas las combinaciones de datos que se hayan establecido en la configuración “Position Configuration”.

4.3.4. Zona de configuraciones

En esta zona se muestran dos paneles de configuración distintos, a los cuales se accede desde la barra de menú. En ellos se introducen los datos básicos para llevar a cabo la simulación. Son el “LiDAR Configuration” (figura 20) y el “Position Configuration” (figura 23).

El primer parámetro del panel de “LiDAR Configuration” es el “Number of planes”, traducido como número de planos en el eje vertical. Es un número natural y su mínimo es 1 puesto que de ser 0 no existiría sensor. La casilla seleccionable de “Planes in regular angles” afecta a los ángulos de los planos. Si “Number of planes” es igual a 15 planos al estar marcado el seleccionable se distribuyen de manera regular entre el mínimo y el máximo. Cuando esto sucede se bloquean los cambios de los planos y se distribuyen siguiendo la siguiente ecuación:

$$\theta = \frac{\theta_{max} - \theta_{min}}{NumPlanes+1} \qquad (4.3.3.1)$$

Los siguientes dos parámetros son el “Vertical Angle” máximo y mínimo y sirven para definir los límites verticales. Están limitados entre -90º y 90º y de ellos se obtiene todo el espectro vertical. Además, están limitados para obligar que el máximo tenga que ser mayor que el mínimo. Los planos también se limitan acorde a los valores introducidos en estos dos *spinbox.* Si uno de los dos variara cuando se ha marcado la casilla de “Planes in regular angles” haría que se recalcularan todos los planos siguiendo la ecuación (4.3.3.1)

El ángulo de “Azimuth” es la resolución horizontal, es decir, la separación entre un haz laser y el siguiente. La “Horizontal Aperture” es la apertura horizontal máxima del sensor, que está limitada entre 0 y 360º. Para saber el número de puntos en un plano habría que dividir la apertura horizontal máxima del sensor entre el ángulo de “Azimuth”.

A través del “Maximum distance” se establece la distancia máxima a la que llega cada haz de láser desde el sensor. Se hace necesario este límite para que, al llevar a cabo la simulación, exista un límite en el que si no se encuentra un obstáculo se detenga la simulación y continúe con el siguiente haz. Además, se consigue añadir un componente de realidad porque un láser no lograría mantener su haz hasta el infinito al ir perdiendo potencia. La distancia predeterminada son 30 000 unidades.

Como en un entorno real existen interferencias que afectan a los láseres, el programa incorpora el parámetro de “Noise level”. Que el valor del ruido sea 0 significa que se va a realizar una simulación ideal sin ruido, pero establecer un valor a este parámetro

equivaldría a la existencia de ruido recorriendo todos los puntos del modelo simulado, añadiéndose ruido gausiano a todas sus coordenadas (x, y, z).

El último apartado dentro del menú "LiDAR Configuration" es la etiqueta de "Plane angle". En ella se encuentra una barra de desplazamiento con *spinboxes* numeradas del 1 hasta el número de planos que se haya fijado. En el caso de que se haya marcado la casilla de "Planes in regular angles" estos *spinboxes* no podrán ser modificados y contendrán el valor que se haya calculado para cada uno. Si se desmarca esta opción se podrían editar los valores y establecer los que se crea preciso. No es necesario que estén en ningún orden concreto, si bien es aconsejable que sean introducidos de menor a mayor o viceversa para que no haya errores a la hora de meter los datos en la aplicación.

Para cambiar de panel de configuración desde el menú "LiDAR Configuration" al de "Position Configuration" (figura 23) es necesario ir a la barra de menú y en la pestaña de LiDAR seleccionar la configuración que se desea ver cómo se puede observar en la figura 18.

En este panel hay dos tipos de información, en uno se encuentra la información espacial relativa a las coordenadas "x", "y" y "z" y a continuación están los datos de los ángulos *roll*, *pitch* y *yaw*. En la figura 22 se pueden ver los ángulos y los ejes de coordenadas.

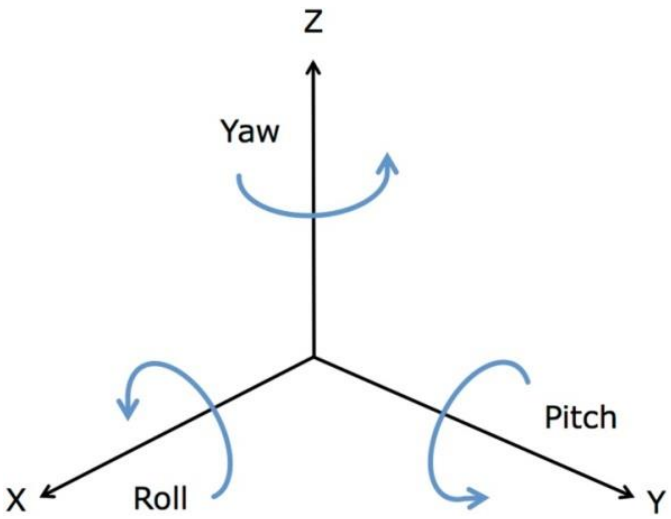

Ilustración 22 Sistema de coordenadas y ángulos [30]

Ambos tipos de información tienen la misma estructura, un apartado de "step" y luego dos *spinbox* para cada coordenada/ángulo. Los "step" son el intervalo entre simulación y simulación, es decir si se van a llevar a cabo simulaciones con el sensor desde 10 a 20 en el eje X y el "step" en X es de 1, se situará el sensor en X igual a 10, 11…19, 20 y siendo igual para las diferentes coordenadas y ángulos. Si cualquiera de los intervalos se fija en 0, que es el valor mínimo y el valor predeterminado de esos parámetros, únicamente se llevará a cabo una simulación en esa coordenada. Dicha simulación efectuará donde esté fijado el mínimo de la coordenada cuyo valor de intervalo es 0. Por ejemplo, en la figura

23 todos los “step” están a 0 por tanto sólo se realizará una simulación en los mínimos de cada coordenada y ángulos.

Cabe destacar que estas coordenadas son para el sensor y que el modelo cargado se situará siempre en el punto (X, Y, Z) = (0, 0, 0).

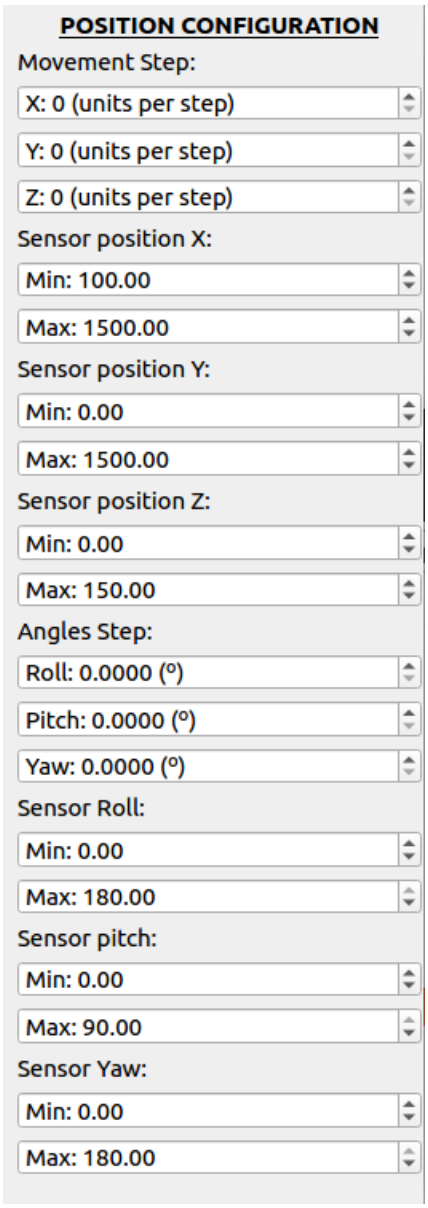

Figura 23 Panel de configuración espacial

5. PROCESO DE SIMULACIÓN

Tras exponer la situación actual en el campo hacia el que va dirigido este proyecto, las herramientas utilizadas en su desarrollo y presentar las funciones de la aplicación, en este apartado se tratará el proceso de simulación, detallándose cada una de las distintas fases necesarias para completar la simulación.

5.1. Fase previa

Una vez el usuario hace click en el botón de "Start Simulation" se inicia la fase previa a la simulación, apareciendo un aviso que informa del inicio del proceso. El aviso se puede ver tanto por línea de comandos como debajo de los botones "Previous" y "Next". A continuación, se realiza un autoguardado del proyecto y las configuraciones del sensor para tener una copia de seguridad por si se diese el caso de un fallo en la aplicación y el programa se cerrara inesperadamente. También se inicializa la nube de puntos temporal que servirá posteriormente como resultado de la fase de simulación.

5.2. Fase preparatoria

En el caso de estar marcada la casilla de "Simulate all models" se iniciarán tantas fases preparatorias como modelos haya abiertos.

En esta fase se vuelve abrir el modelo guardado, pero, esta vez en vez de abrir sus puntos y representarlos gráficamente, se abren dichos puntos unidos en triángulos, no siendo representados en pantalla para optimizar los recursos computacionales. El programa comprueba que el número de puntos no sea igual a cero, puesto que, de ser cero, se pararía la ejecución de la simulación de inmediato al no tener el modelo ningún punto. No siendo cero continuaría el proceso consultando la hora local para prepararar el modelo de distribución gausiano aleatoriamente. Se inicializa la variable de posición del sensor en X (a partir de ahora posición en X) y se le asigna el valor mínimo para dicha coordenada que previamente se ha proporcionado en el panel de configuración de posición, fijándose el valor por defecto si no se hubiese asignado ese valor mínimo. También se obtienen los límites del modelo para poder descartar los puntos que se salgan de esos límites, ahorrando así tiempo de procesamiento. Aquí se inicializa también el contador de número de archivo.

5.3. Fase de muestreo espacial y angular

Para comprender el funcionamiento del programa en esta fase hay que entender los parámetros de posición representados en la figura 20, puesto que será preciso ir cambiándolos hasta obtener todas las combinaciones posibles. Para mejorar la comprensión de este apartado, En la figura 24 se encuentra un diagrama de bloques del funcionamiento de esta fase.

En primer lugar, se crea un bucle que va desde la posición mínima de X hasta la máxima en pasos de valor "stepX". Para todos los valores de X se crea un bucle con los valores de Y, del mínimo al máximo, encontrándose a su vez, dentro de cada valor, otro bucle

con todos los valores de Z del mínimo al máximo en pasos de “stepZ”. Y así se tienen muestreados todos los valores de “X”, “Y” y “Z” del sensor en el espacio.

Para cada valor espacial se aplican también bucles de ángulos. Es decir, se recorren todos los valores de *roll* desde el mínimo al máximo en variaciones de “stepRoll”. Como el *roll* es el ángulo de giro del eje X, esto se representa como un ángulo del plano “YZ” así que para cada ángulo de *roll* se calcula su seno y su coseno y se aplican a los ejes “Y” y “Z”. Una vez hecho eso se inicia un bucle para obtener todos los valores de pitch, generándose un bucle con los valores de *yaw* para cada valor de *pitch*.

Cada vez que se obtiene un valor distinto de cada uno se lleva a cabo de nuevo la fase de simulación, que una vez terminada vuelve a iniciarse con el siguiente grupo de valores.

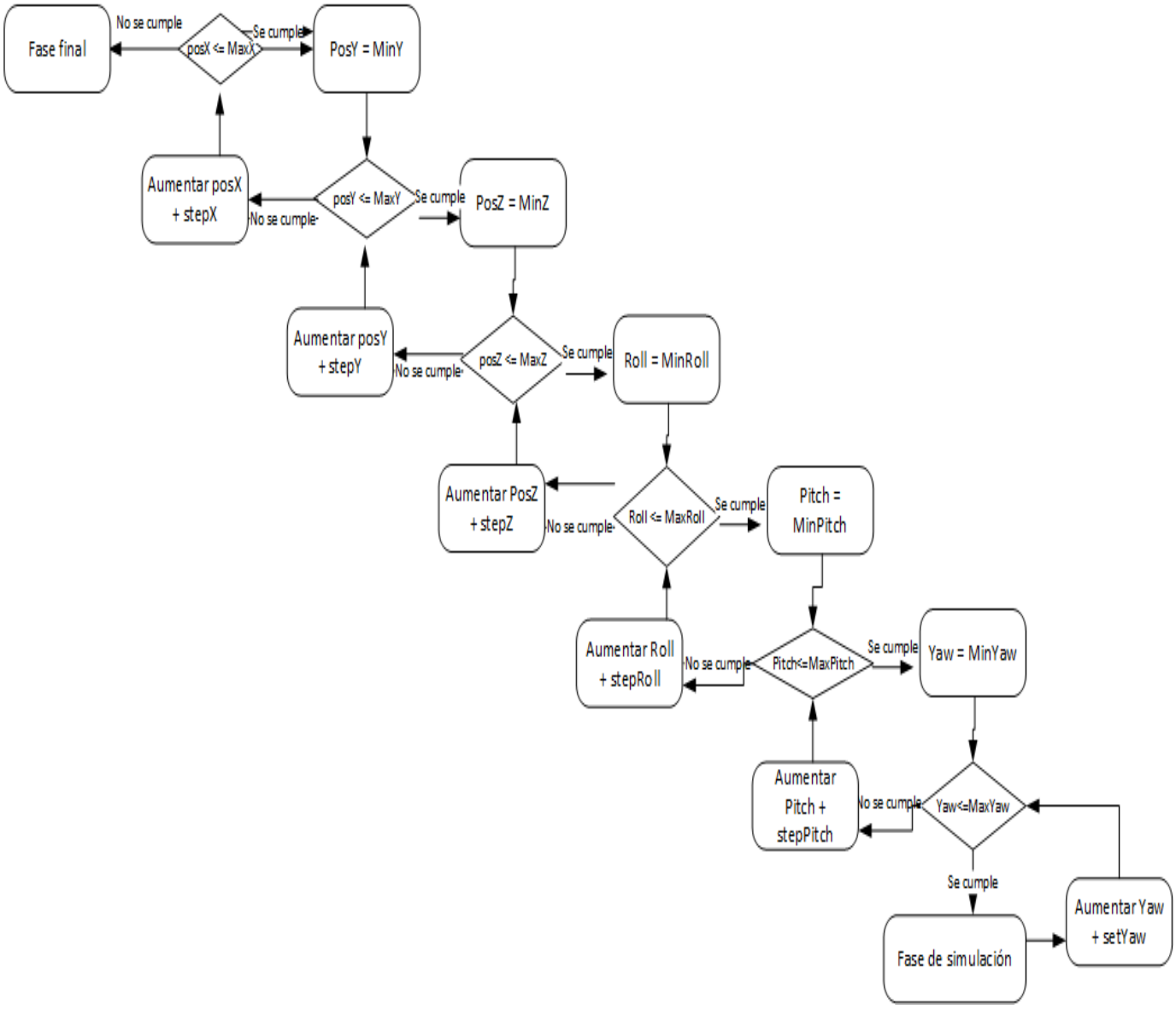

Figura 24 Diagrama de flujo de la fase de muestreo espacial y temporal

5.4. Fase de simulación

Esta fase es en esencia un trazado de rayos para, posteriormente, intersectarlos con las superficies visibles de los modelos a simular.

La fase comienza vaciando la nube de puntos temporal y comprobando que el sensor no se encuentra en el mismo punto que el modelo, de ser así se imprimiría un error por línea de comandos y se proseguiría con el siguiente punto en el que deba encontrarse el sensor. Posteriormente se sitúa un sistema de referencia móvil en el sensor y se realizan las inclinaciones correspondientes.

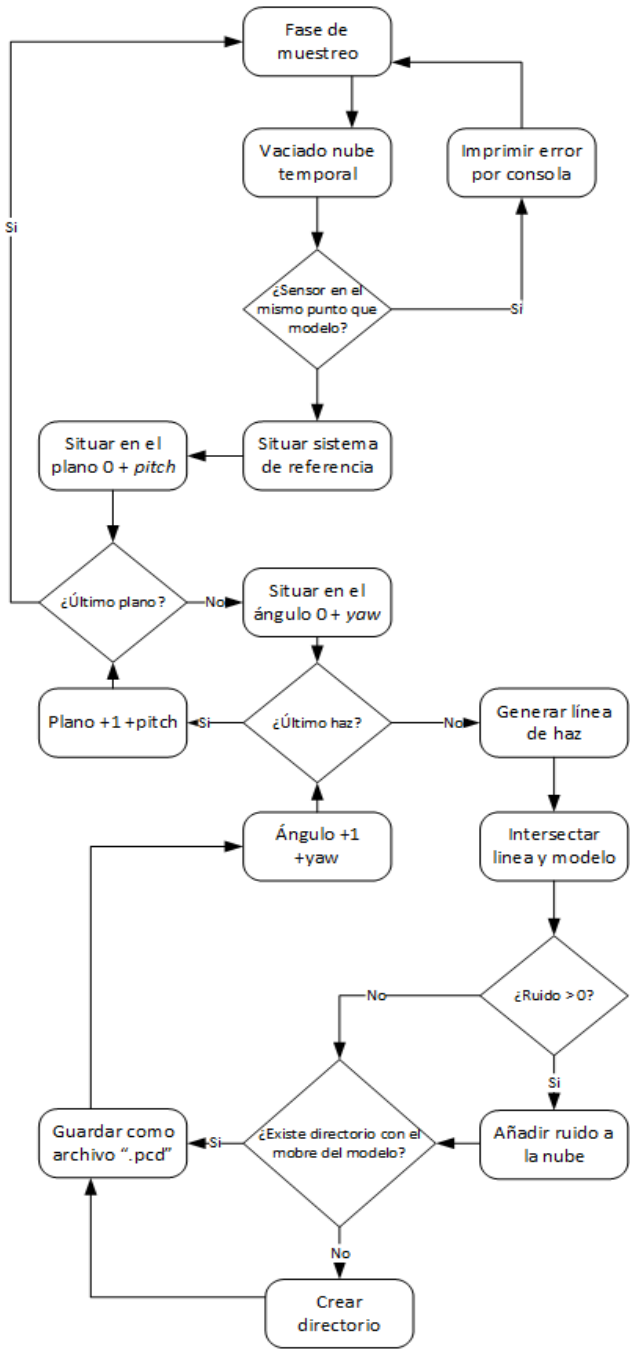

Figura 25 Diagrama de flujo de la fase de simulación

Una vez que concretado el sistema de referencia se inicia el barrido vertical de planos. Se empieza por el plano número 1 y se incrementan de uno en uno hasta alcanzar el último. A cada ángulo vertical se le suma el ángulo de *pitch* que toque en su simulación. Una vez completado el barrido vertical se inicia el barrido horizontal de haces. A cada ángulo horizontal se le suma el ángulo de *yaw* correspondiente y se continúa rotando hasta alcanzar la apertura máxima del sensor.

Se traza un rayo virtual con lo que sería el haz del LiDAR y posteriormente, se calcula la intersección de cada haz con el modelo. Dicha intersección se calculará de manera diferente para cada modelo dependiendo de su formato de entrada. El resultado de la operación anterior es un punto que se añade a la nube de puntos temporal. El resultado del cálculo de todas las intersecciones del modelo con los haces sería la nube de puntos generada que representa el resultado ideal sin interferencias ni variaciones.

Para conseguir resultados acordes a la de un entorno real se añade ruido a todas las coordenadas de los puntos en la nube de puntos generada. En el caso de que el nivel de ruido se haya asignado a cero, se ignorará este paso y pasará al siguiente. El ruido se genera sumándole el resultado de la ecuación 5.4.0.2. (distribución normal) para una x aleatoria entre 0 y 1 y una σ igual al nivel de ruido seleccionado.

$$Ruido = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} e^{-\frac{x}{2\sigma^2}} \quad (5.4.0.2.)$$

Una vez comprobada la no existencia de puntos duplicados en la nube se crea, en caso de no existir, un directorio bajo el mismo nombre que el modelo en caso de no existir ya y se guarda la nube de puntos con el siguiente formato: número de simulación + "X" + coordenada X del sensor + "Y" + coordenada Y del sensor + "Z" + coordenada Z del sensor + "R" + ángulo de *roll* + "P" + ángulo de *pitch* + "W" + ángulo de *yaw* + ".pcd". De esta manera aparecen las simulaciones ordenadas y dando información de los parámetros con los que se han generado.

Una vez guardado el archivo se aumenta el número de simulación y se continúa en la siguiente simulación con un grupo de parámetros distintos.

5.5. Fase final

Para terminar, se imprime por consola el número de simulaciones realizadas y se carga la primera simulación generada al visualizador. Se cambia el mensaje de aviso a simulación terminada y se vuelve a la ejecución normal del programa.

6. RESULTADOS

Para explicar el funcionamiento de los diversos parámetros de configuración disponibles en la aplicación, se ha determinado analizar cada uno de ellos por separado, introduciendo variaciones mientras los demás se mantienen constantes. En las distintas figuras presentadas, el modelo real de color azul, y el modelo simulado, resultante del proceso de trazado de rayos, de color rojo. Las simulaciones se han realizado con todos los modelos indicados en el apartado 6.1, sin embargo, se ha decidido publicar en esta memoria únicamente los resultados del "cubo de basura" puesto que es un modelo lo suficientemente complejo como para ver cambios en el sensor, pero lo adecuadamente simple como para distinguirlo correctamente en una sola imagen.

6.1. Prueba de distintos modelos

Con la misma configuración y posición del sensor se han simulado distintos modelos para verificar que es consistente con distintos tipos de modelos. Hay que destacar que los modelos están a distintas escalas y por tanto la distancia hasta el sensor y la separación de los planos en cada uno de ellos pueden aparecer distintas.

Figura 26 Simulación de avestruz

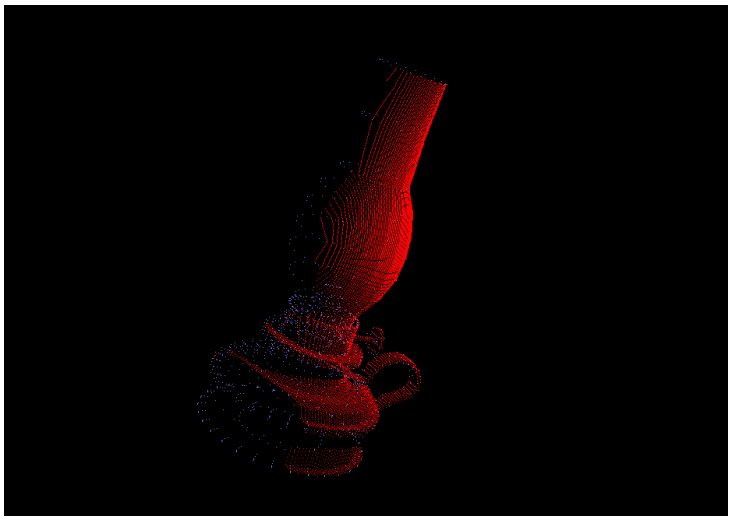

Figura 27 Simulación de lámpara

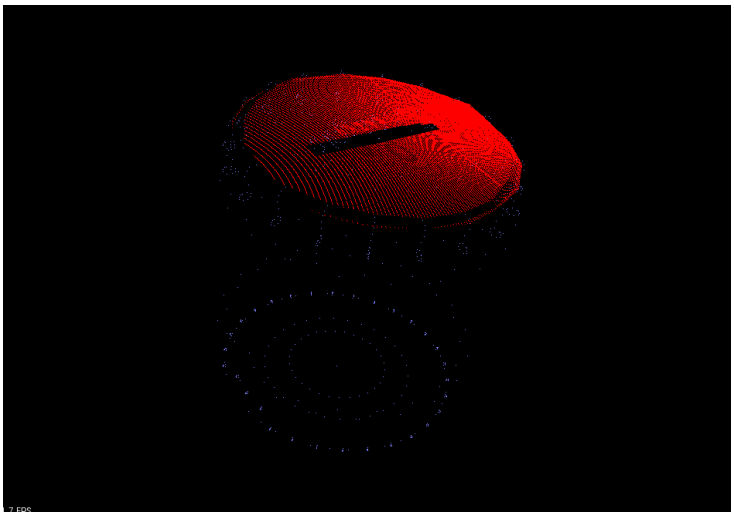

Figura 28 Simulación de cubo de la basura

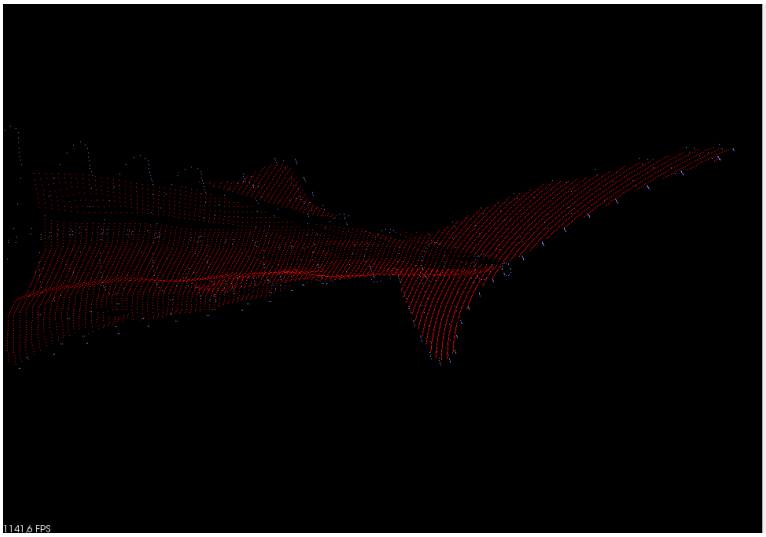

Figura 29 Simulación de aleta de tiburón

6.2. Prueba de "Number of Planes"

Se puede observar que cuanto mayor es el número de planos más puntos se concentran en el eje vertical.

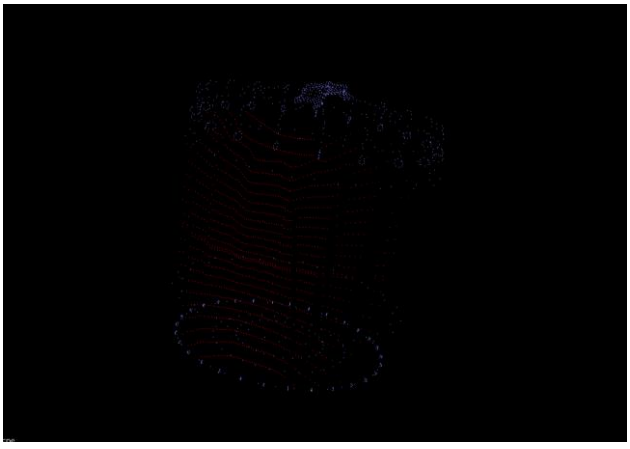

Figura 30 Nube de puntos generada para un LiDAR virtual de 50 planos

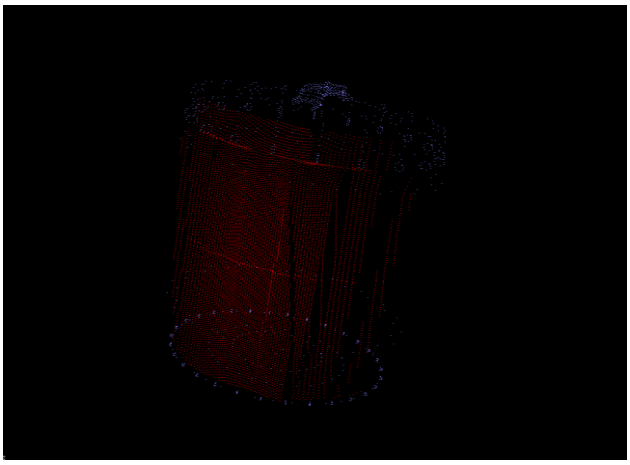

Figura 31 Nube de puntos generada para un LiDAR virtual de 100 planos

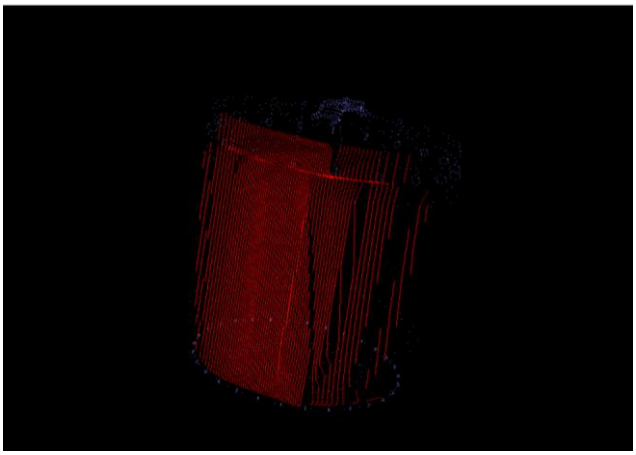

Figura 32 33Nube de puntos generada para un LiDAR virtual de 300 planos

6.3. Prueba de "Vertical angle"

Para esta prueba se han variado el ángulo máximo y mínimo dejando el número de planos constantes y ajustados automáticamente de manera regular para ver cómo afecta una apertura vertical mayor o menor sobre el modelo.

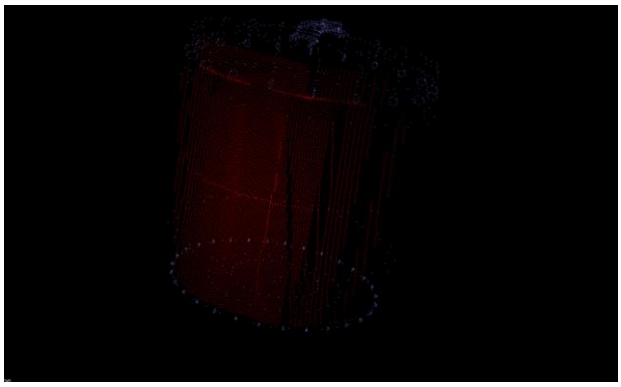

Figura 34 Nube de puntos generada para un LiDAR virtual de apertura mínima 90º y apertura máxima 90º

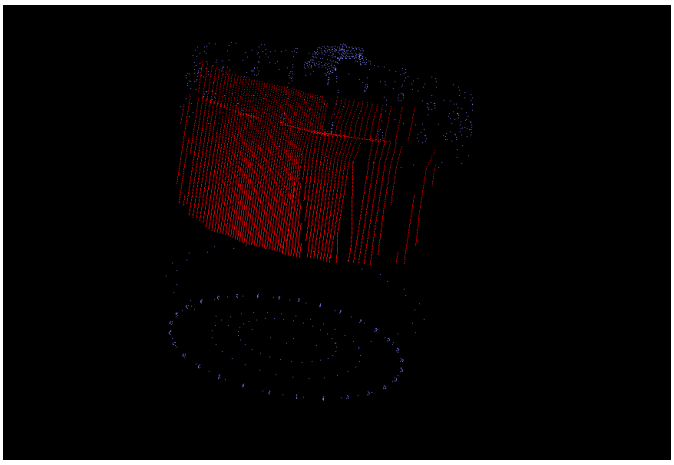

Figura 35 Nube de puntos generada para un LiDAR virtual de apertura mínima 90º y apertura máxima 0º

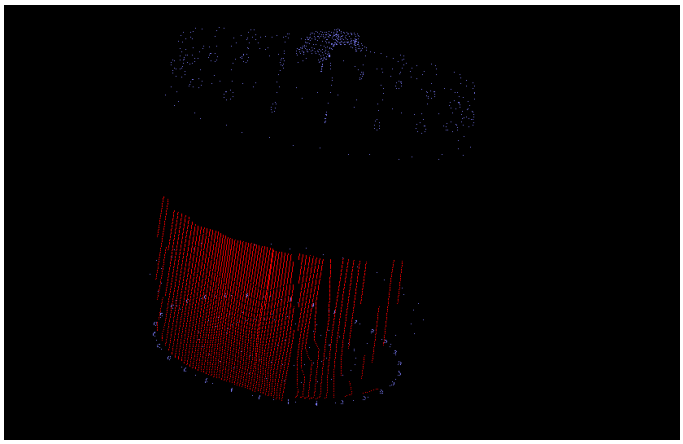

Figura 36 Nube de puntos generada para un LiDAR virtual de apertura mínima 0º y apertura máxima 90º

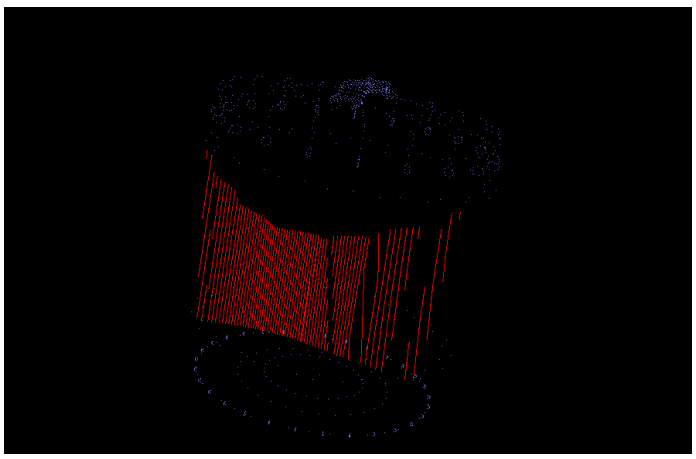

Figura 37 Nube de puntos generada para un LiDAR virtual de apertura mínima 20º y apertura máxima 20º

6.4. Prueba de “Azimuth”

La variación del “Azimuth” trae consigo una modificación en los puntos horizontales, a menor “Azimuth” mayor número de puntos y viceversa.

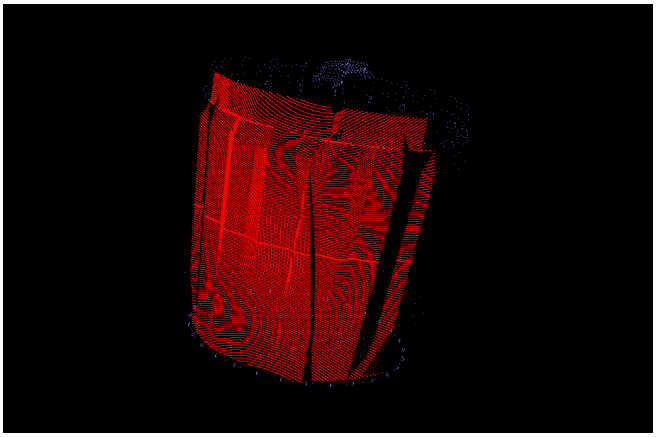

Figura 38 Nube de puntos generada para un LiDAR virtual de 0.01º de “Azimuth”

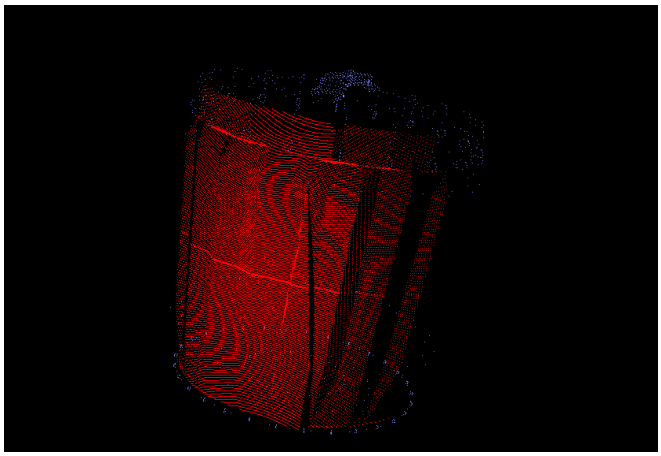

Figura 39 Nube de puntos generada para un LiDAR virtual de 0.25º de “Azimuth”

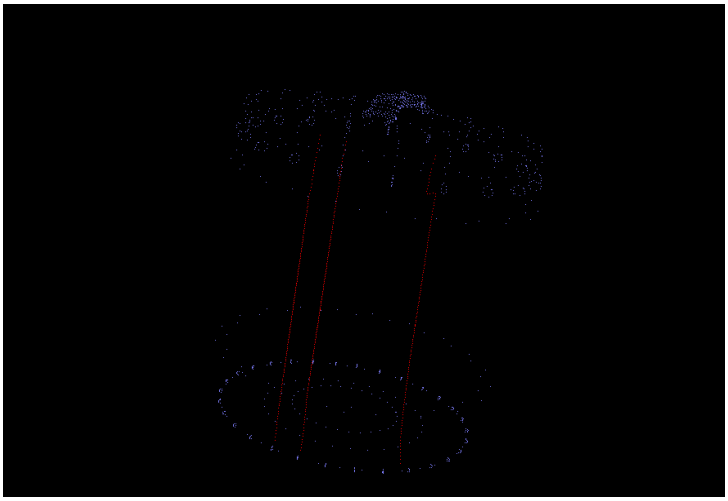

Figura 40 Nube de puntos generada para un LiDAR virtual de 25º de “Azimuth”

6.5. Prueba de “Horizontal Aperture”

Para esta prueba se ha disminuido la apertura del ángulo con el sensor centrado en el cubo y se observa la reducción de puntos en el eje horizontal.

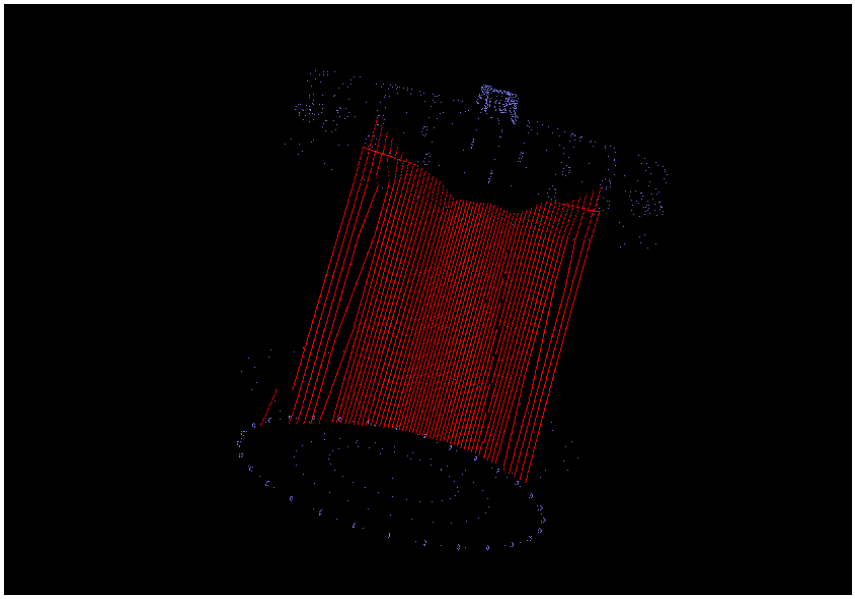

Figura 41 Nube de puntos generada para un LiDAR virtual de 50º de "Horizontal Aperture"

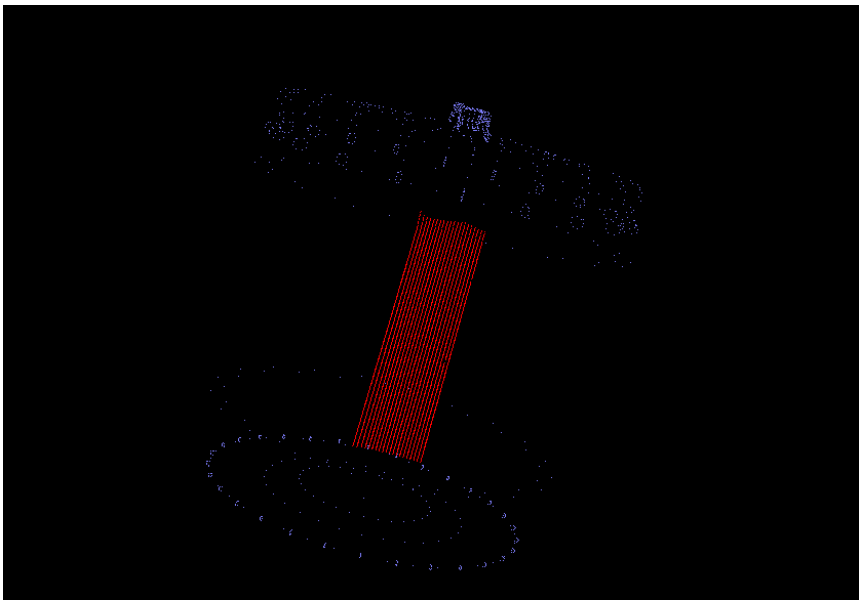

Figura 42 Nube de puntos generada para un LiDAR virtual de 20º de "Horizontal Aperture"

6.6. Prueba de “Noise level”

Para esta prueba, con un interés centrado en la mejor visualización, se ha elegido la vista en la que el sensor genera líneas horizontales para que se vea mejor la variación que produce el ruido. En la figura 43 podemos observar cómo quedaría sin ruido y en las figuras siguientes se puede ver cómo aumentando el ruido empeora la visión de las líneas paralelas.

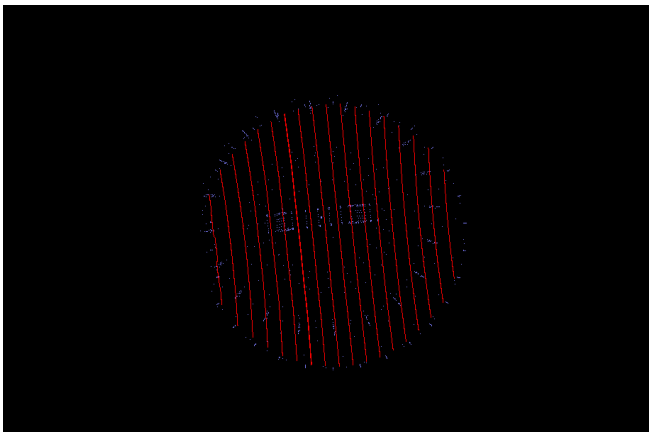

Figura 43 Nube de puntos generada para un LiDAR virtual sin ruido

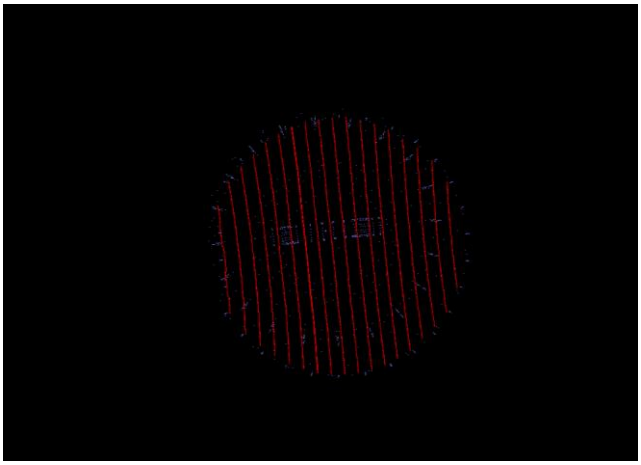

Figura 44 Nube de puntos generada para un LiDAR virtual con ruido de 0.15

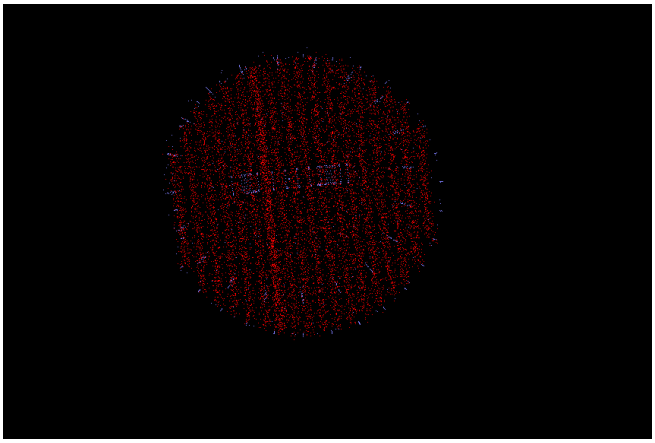

Figura 45 Nube de puntos generada para un LiDAR virtual con ruido de 0.75

6.7. Prueba de coordenada X

Para esta prueba se han configurado los parámetros espaciales a 0 y se ha variado únicamente la componente “X”. Se puede observar como con una “X” negativa el sensor se encuentra por detrás del cubo y, con una “X” positiva se encuentra delante. Así mismo se aprecia que al estar más lejos, los planos del sensor aparecen más separados entre sí.

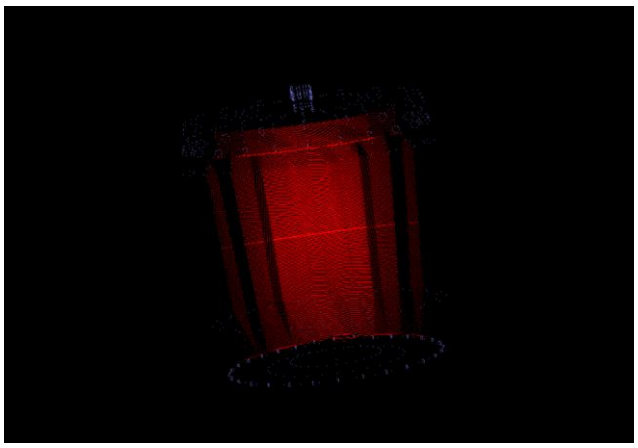

Figura 46 Nube de puntos generada para un LiDAR virtual situado en X = -50

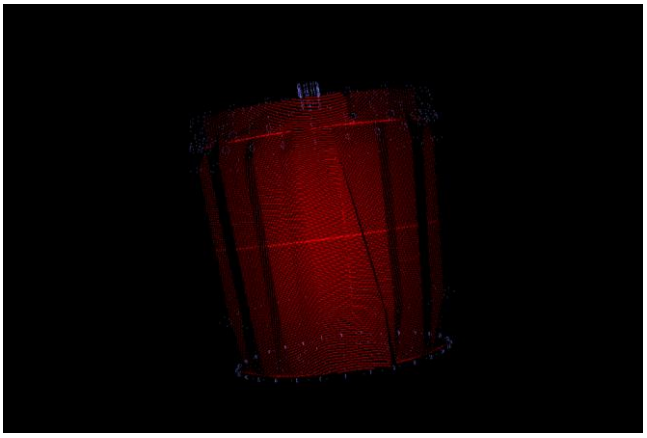

Figura 47 Nube de puntos generada para un LiDAR virtual situado en X = 50

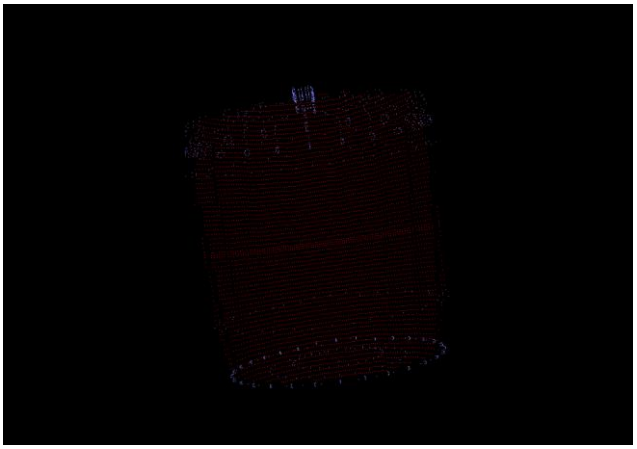

Figura 48 Nube de puntos generada para un LiDAR virtual situado en X = 100

6.8. Prueba de coordenada Y

Utilizando el mismo procedimiento que en la prueba de coordenada “X”, se han configurado los parámetros espaciales a 0 y solo se ha variado la componente “Y”. Se puede observar que con una “Y” negativa el sensor se encuentra a la izquierda del cubo y con una “Y” positiva se encuentra a la derecha. Así mismo se aprecia que debido a la diferencia en la distancia los planos del sensor aparecen más separados entre sí.

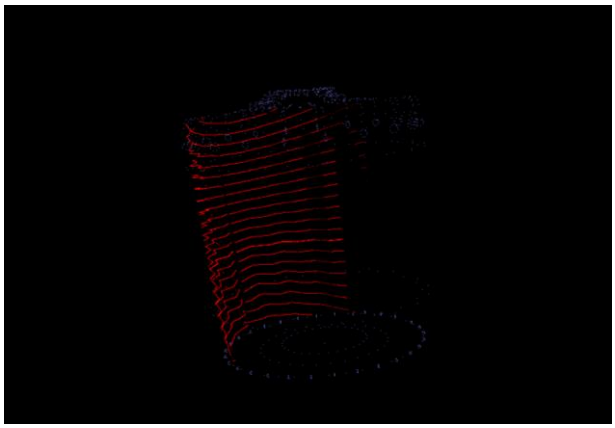

Figura 49 Nube de puntos generada para un LiDAR virtual situado en Y = -100

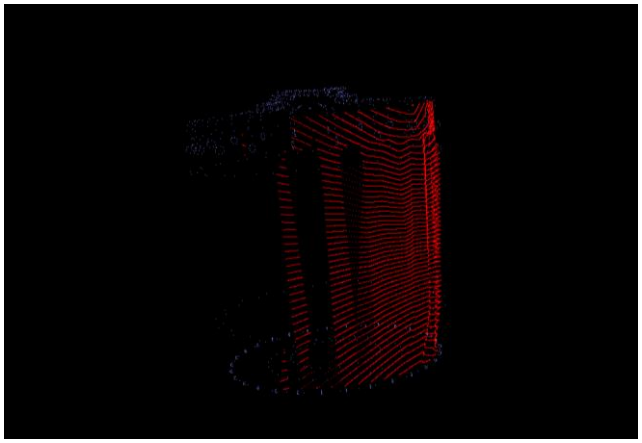

Figura 50 Nube de puntos generada para un LiDAR virtual situado en Y = 50

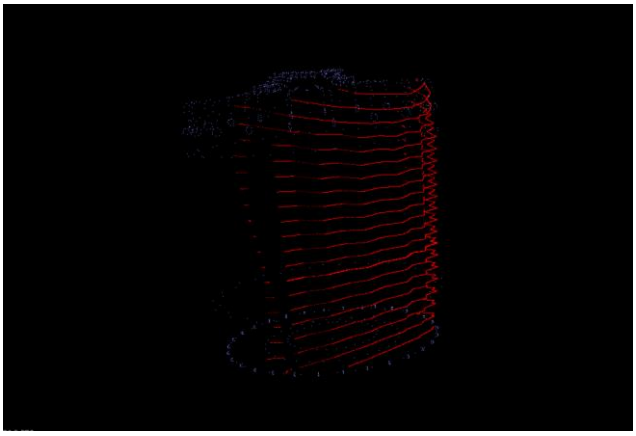

Figura 51 Nube de puntos generada para un LiDAR virtual situado en Y = 100

6.9. Prueba de coordenada Z

De nuevo, siguiendo el procedimiento que ya hemos visto para la “X” y la “Y”, se han dispuesto los parámetros espaciales a 0, modificando únicamente la componente “Z”. Se comprueba que con una “Z” negativa el sensor se encuentra debajo del cubo y con una “Z” positiva se encuentra por encima. También se aprecia que, al estar más alejado, los planos del sensor aparecen más separados entre sí.

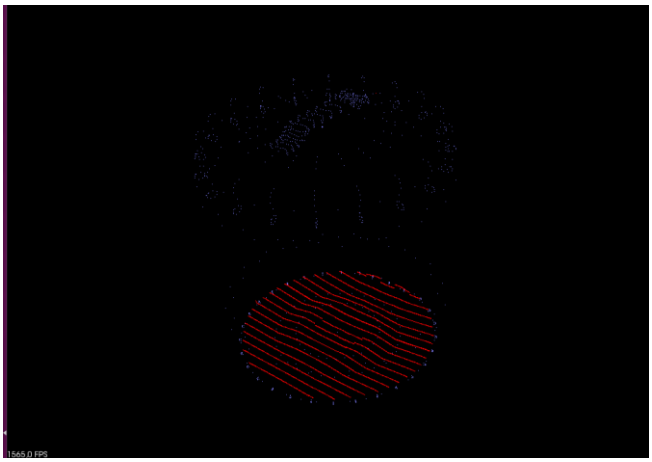

Figura 52 Nube de puntos generada para un LiDAR virtual situado en Z = -100

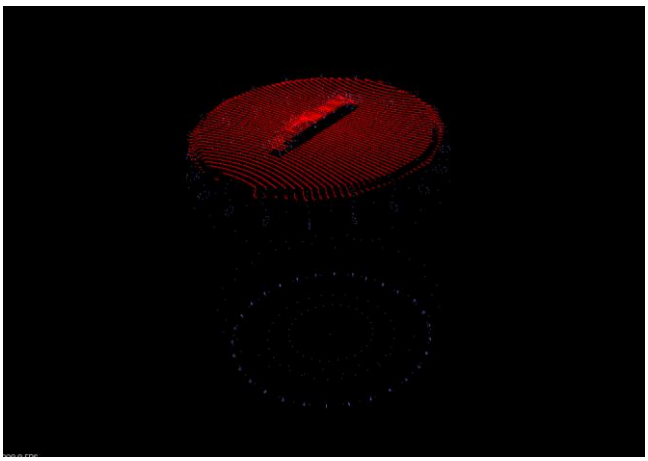

Figura 53 Nube de puntos generada para un LiDAR virtual situado en Z = 50

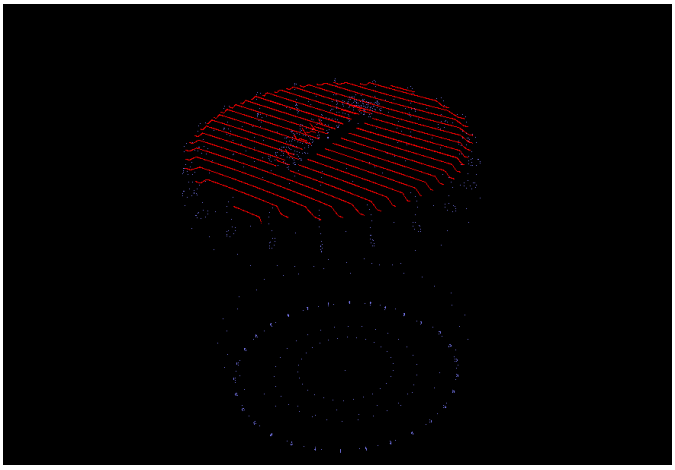

Figura 54 Nube de puntos generada para un LiDAR virtual situado en Z = 100

6.10. Prueba de ángulo de “Roll”

El objetivo de esta prueba comprobar cómo varía la inclinación. Para ello se ha utilizado un sensor centrado en el cubo y con una apertura vertical de 20º. En la figura 49 se observa como las líneas son verticales, mientras que en las dos figuras la inclinación de las líneas varía en 45º a un lado o al otro.

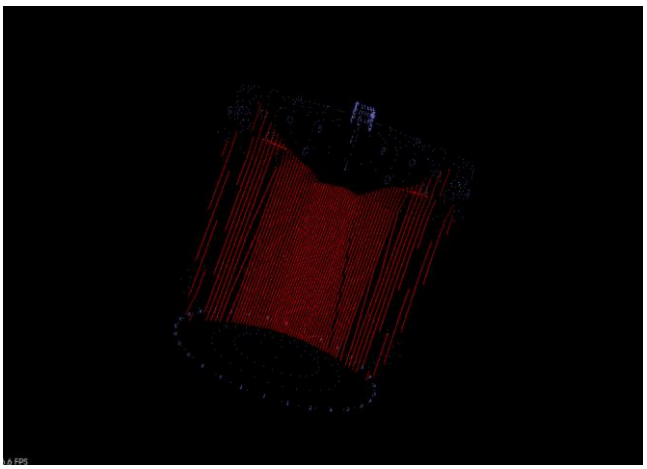

Figura 55 Nube de puntos generada para un LiDAR virtual con un "Roll" de 0º

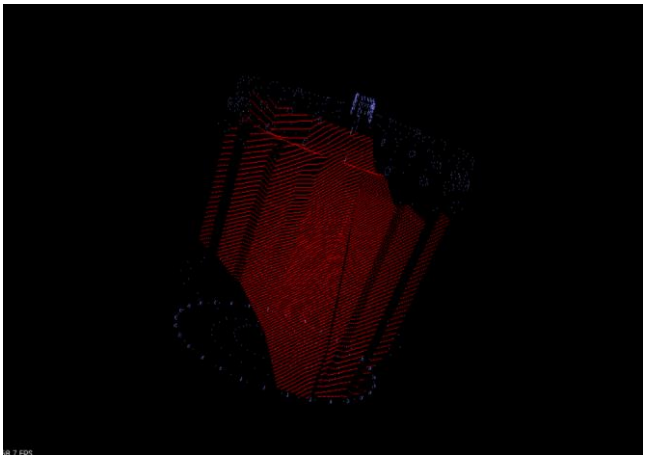

Figura 56 Nube de puntos generada para un LiDAR virtual con un "Roll" de 45º

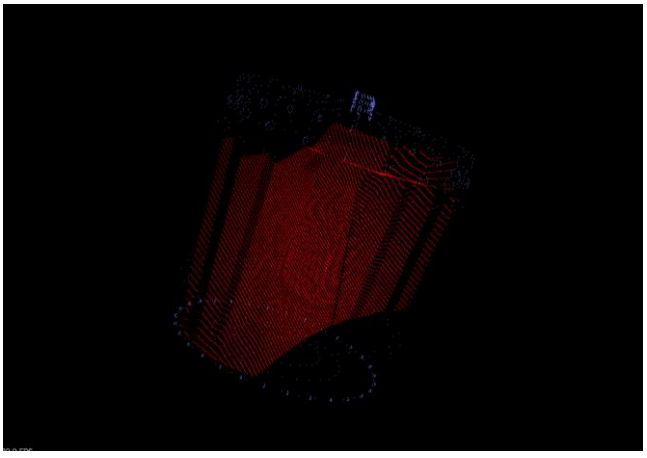

Figura 57 Nube de puntos generada para un LiDAR virtual con un "Roll" de -45º

6.11. Prueba de ángulo de “Pitch”

Como este ángulo mide la inclinación en el eje “Y”, el efecto que produce es que levanta o baja el espectro del sensor. Con un “Pitch” negativo sube y con uno negativo baja.

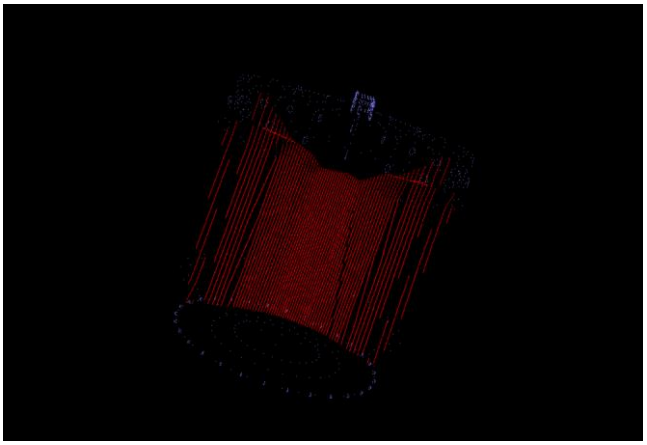

Figura 58 Nube de puntos generada para un LiDAR virtual con un "Pitch" de 0º

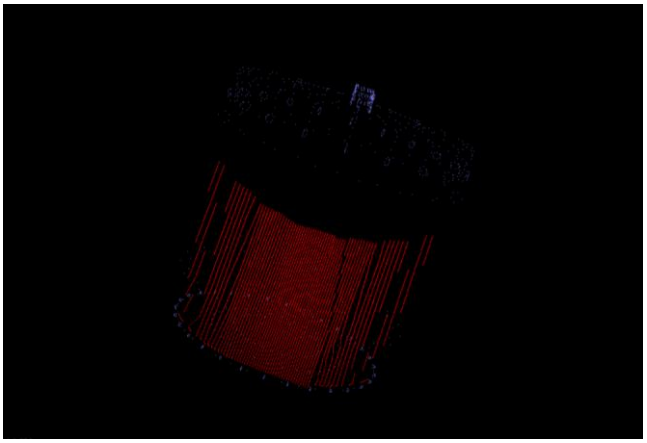

Figura 59 Nube de puntos generada para un LiDAR virtual con un "Pitch" de 25º

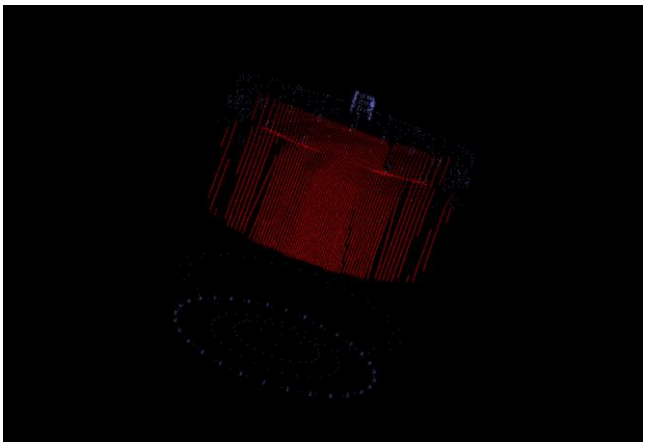

Figura 60 Nube de puntos generada para un LiDAR virtual con un "Pitch" de 25º

6.12. Prueba de ángulo de “Yaw”

Con un sensor con una apertura limitada podemos observar que, si se modifica el “Yaw”, tenemos un desplazamiento horizontal del sensor. Los ángulos negativos desplazan el haz del sensor a la izquierda y los positivos a la derecha.

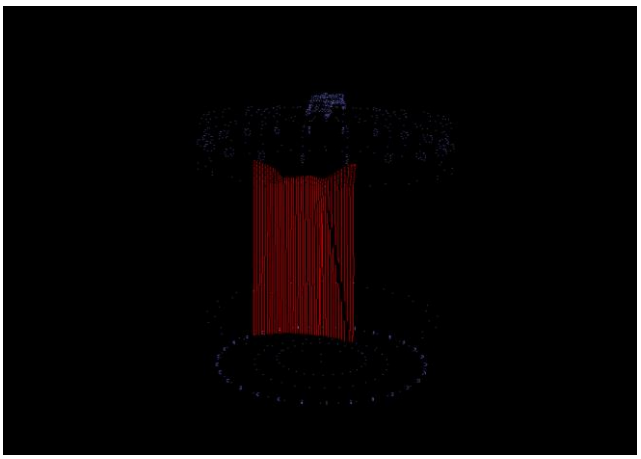

Figura 61 Nube de puntos generada para un LiDAR virtual con un "Yaw" de 0º

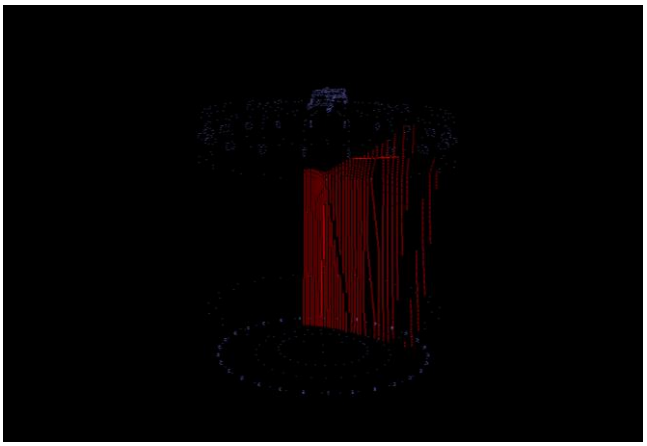

Figura 62 Nube de puntos generada para un LiDAR virtual con un "Yaw" de 20º

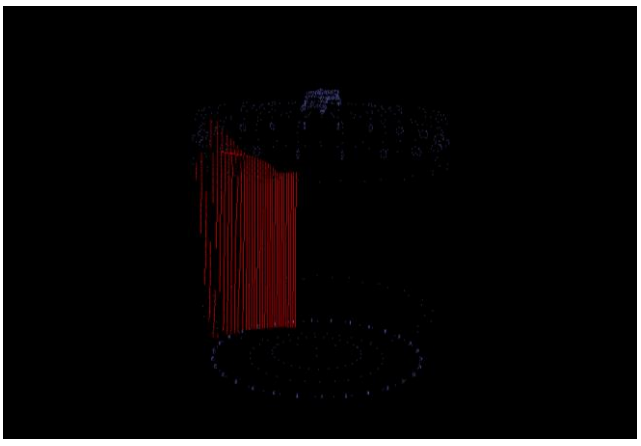

Figura 63 Nube de puntos generada para un LiDAR virtual con un "Yaw" de -20º

7. PRESUPUESTO

En el apartado de presupuesto se presentan los cálculos los costes materiales y personales que han sido precisos asumir para la elaboración de este proyecto.

Durante la elaboración de este desarrollo se han precisado únicamente de dos elementos que podríamos englobar en los costes en recursos materiales. Se trataría de dos ordenadores, uno portátil y otro de sobremesa. El coste del portátil es de 880 euros con las mejoras que incluía y el precio del ordenador personalizado de sobremesa es de 1200 euros. El coste para el proyecto no es el valor total de los recursos, sino que hay que aplicarle un coeficiente de amortización que viene dado por la Agencia Tributaria y que puede consultarse en su web [31]. Para este tipo de material el índice de amortización es de 26% anual, lo que equivaldría a 2.17% mensual.

TABLA 1 COSTES MATERIALES

ELEMENTO	PRECIO (€)	COEFICIENTE DE AMORTIZACIÓN	TIEMPO (meses)	COSTE (€)
Portátil	880	0,0217	4	76,27
Ordenador de mesa	1200	0,0217	3	78,00

Los costes en recursos personales son los que generan el sueldo de un ingeniero electrónico con menos de 5 años de experiencia, salario que tiende a oscilar en torno a los 30.000 euros anuales brutos. [32]

TABLA 2 COSTES PERSONALES

SUELDO BRUTO ANUAL (€)	TIEMPO (meses)	COSTE (€)
30.000,00 €	7	17.500,00 €

El coste total será la suma de costes en los recursos personales y materiales que han sido precisos para elaborar este proyecto que asciende a 17.654,27€.

8. MARCO REGULADOR

Para poder circular con un vehículo autónomo en España es necesario un permiso especial expedido por la DGT (Dirección General de Tráfico), tanto si se llevan las funciones de conducción autónomas activadas o desactivadas. Dicho permiso ese temporal y únicamente para la realización de pruebas o investigación.

La DGT define vehículo autónomo como *"Todo vehículo con capacidad motriz equipado con tecnología que permita su manejo o conducción sin precisar la forma activa de control o supervisión de un conductor, tanto si dicha tecnología autónoma estuviera activada o desactivada, de forma permanente o temporal."* También añade *"Son objeto de esta instrucción aquellos vehículos que incorporan tecnología con funciones asociadas a los niveles automatización 3,4 y 5"*. [33]

Los niveles son los siguientes:

- Nivel 0: No cuenta con tecnología de conducción autónoma y es el conductor el que realiza todas las labores.
- Nivel 1: Conducción asistida. El vehículo ayuda en tareas concretas, como podría ser mantener la distancia de seguridad.
- Nivel 2: Conducción parcialmente automatizada. El vehículo puede circular en ciertos entornos definidos bajo la atenta supervisión del conductor que podrá y deberá tomar el control en cualquier momento.
- Nivel 3: Conducción automatizada condicionada. El vehículo puede circular en ciertos entornos definidos y reconocer cuando debe ceder el control al conductor con antelación suficiente.
- Nivel 4: Conducción altamente automatizada. El vehículo puede circular en todas las situaciones, incluso cuando el conductor no responde a pesar de que el sistema le indique que tome el control. Sigue siendo obligatoria la presencia del conductor en el vehículo.
- Nivel 5: Conducción plenamente automatizada. El vehículo puede circular en todos los entornos sin necesidad de conductor en el vehículo.

Los requisitos a cumplir para poder obtener un permiso de pruebas son los siguientes:

- Ser fabricante o tener relación directa con el mismo, o pertenecer a la universidad o consorcios que participen en proyectos de investigación.
- Aportar la documentación prevista
- Identificar correctamente el vehículo y tener un seguro para el mismo.
- Acreditar que el vehículo ha superado las pruebas técnicas estipuladas tanto en España como en otro Estado Miembro de la Unión Europea.
- El conductor o conductores estarán identificados y cumplirán con la normativa vigente para la circulación. Teniendo una antigüedad mínima de dos años del permiso de conducción del tipo de vehículo con el que se va a llevar a cabo el ensayo.

9. CONCLUSIONES

Finalizando esta memoria que ha servido para exponer el proyecto llevado a cabo, se hace preciso valorar cualitativa y cuantitativamente el nivel de cumplimiento de los objetivos en el apartado 1.2., objetivos que, por comodidad de lectura, se reproducen a continuación:

1. Conseguir una correcta simulación de un sensor LiDAR.
2. Generar, mediante simulaciones, nubes de puntos desde distintos puntos de vista en distintas condiciones de ruido.
3. Soportar los formatos de archivos más comunes actualmente utilizados para modelos 3D.
4. . Desarrollar un software que facilite y agilice el proceso de creación de bases de datos de nubes de puntos anotadas, automatizando en gran medida el proceso.
5. Crear una base de datos de nubes de puntos a partir de modelos sintéticos.

El primer objetivo se puede dar plenamente como cumplido puesto que las variaciones llevadas a cabo en las pruebas presentadas en el apartado 6 se ajustan a las propias de un sensor LiDAR en un entorno de conducción real.

El segundo objetivo también se puede dar como cumplido al comprobar la iteración de los parámetros de posición del sensor. Estos resultados también están recopilados en el apartado 6.

El tercer objetivo requiere de una comprensión de las condiciones que han rodeado al desarrollo del trabajo para así obtener una visión realista de si ha sido o no cumplido. Mientras exista un tipo de archivo que este programa no abra, siempre podría considerarse que es un objetivo no satisfecho. Sin embargo, y en base a las limitaciones temporales propias de estos tipos de trabajos, se considera que el número de extensiones compatibles es suficiente y adecuado, teniendo en cuenta que habría sido necesario invertir más tiempo en añadir nuevos tipos de modelos que el total destinado a la elaboración del trabajo; por lo que el objetivo se considera como cumplido.

El cuarto objetivo se considera totalmente como cumplido puesto que en la fase de obtención de datos de entrenamiento se reduce drásticamente el tiempo y recursos necesarios al ser completamente automática.

El quinto objetivo también se cumple puesto que se ha generado una base de datos con los objetos simulados en los resultados. Algunos ejemplos se pueden ver en la figura 64.

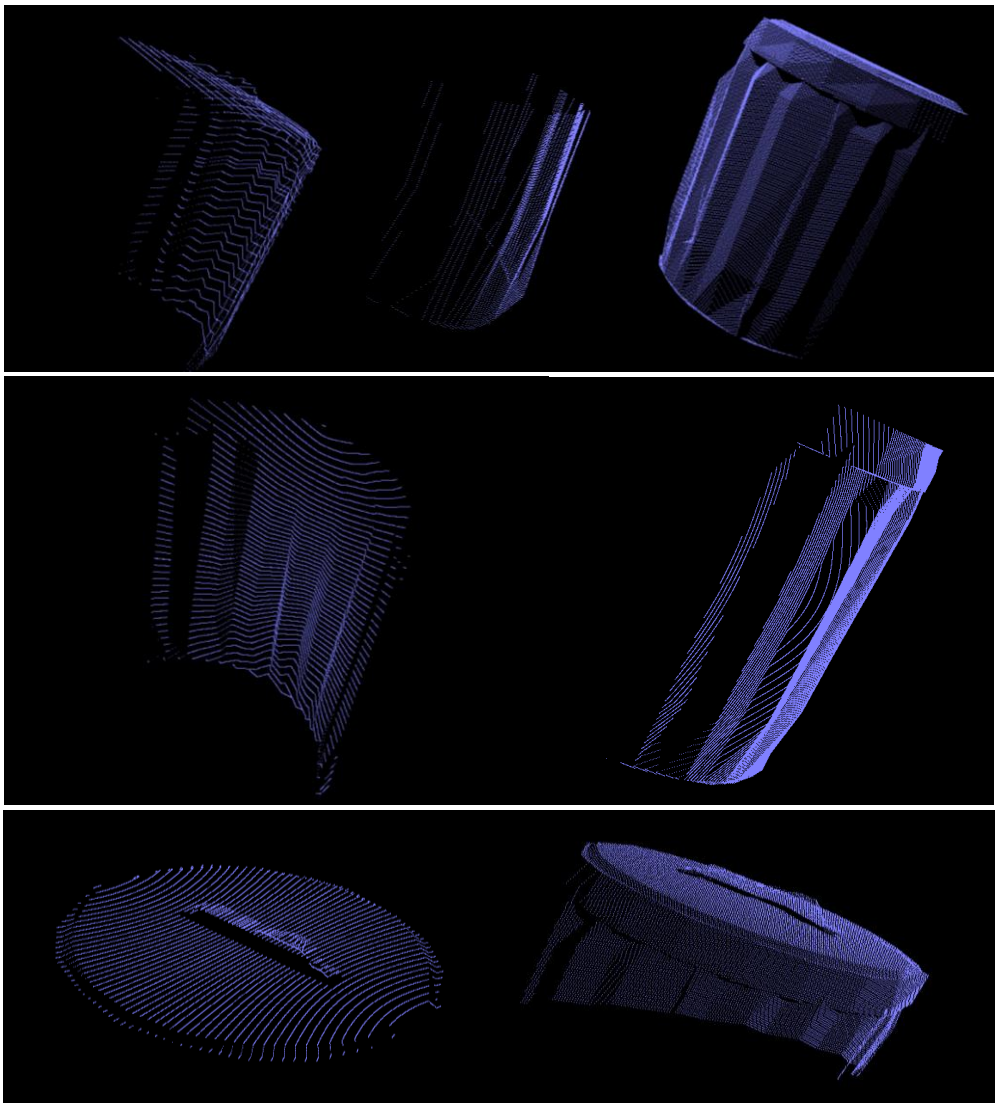

Figura 64 Ejemplos de la base de datos generada

Por tanto, y con los resultados obtenidos, se puede concluir que este programa es funcional y apto para generar bases de datos de nubes de puntos anotadas a partir de modelos sintéticos, logrando un coste temporal bastante bajo, puesto que, si se quisiese generar dichas bases de datos en un entorno real se requeriría de muchos recursos tanto económicos como personales. Gracias a los buenos resultados de este proyecto se pueden utilizar las simulaciones obtenidas para entrenar futuras redes neuronales.

10. TRABAJOS FUTUROS

Una primera vía de trabajo de cara a continuar el desarrollo de la aplicación sería la ampliación del número de tipos de archivos compatibles para la simulación. En la actual versión el programa solo puede trabajar con modelos 3D archivados bajo las extensiones .ply, .vtk y .obj, algo que limita su funcionalidad al no poder acceder a cualquiera de los otros muchos tipos de archivos existentes. Así mismo, se podría trabajar en la mejora de la interfaz gráfica, haciéndola más intuitiva y vistosa puesto que, al ser un aspecto de refinamiento que no afecta directamente en la funcionalidad del programa, no se ha invertido tiempo suficiente en este aspecto.

Un gran avance que mejoraría la usabilidad sería optimizar el tiempo de simulación pasando la aplicación a ser *multicore*. En la actual versión el programa no logra aprovechar todos los procesadores del computador, perdiendo tiempo mientras simula. Sirva de ejemplo que, con el ordenador utilizado en la realización de este proyecto, las simulaciones se podrían realizar hasta ocho veces más rápido.

Otra vía que completaría el producto y lo haría más interesante sería generar una base de datos de automóviles y objetos que se puedan encontrar en un entorno viario para que con ella se pudiese entrenar una red neuronal y comprobar si, mediante la generación de bases de datos, es posible con este programa obtener buenos resultados de clasificación. Así mismo se podría probar el funcionamiento del sensor en un entorno virtual complejo y con dinámicas, por ejemplo, las simulaciones urbanas que se pueden encontrar en algunos videojuegos, para obtener bases de datos complejas.

Como conclusión, realmente la base del desarrollo de esta aplicación está ya completada. Las futuras mejoras podrían incrementar su usabilidad y eficiencia además de concretar su uso en aspectos más específicos con miras a ser de gran utilidad, pero la versión del programa aquí presentada es funcional y capaz de completar las tareas que se le encomienden.

BIBLIOGRAFÍA

[1] F.J. Murillo, "¿Cuáles son los países con más vehículos?", *Expansión* 30-04-2017. **[En línea]. Disponible en:** http://www.expansion.com/economia/2017/04/30/5901b473e5fdea25558b45ad.html

[2] P. Pettit, "World Vehicle Population Rose 4.6% in 2016", *Wards intelligence* 17-10-2017. **[En línea]. Disponible en:** http://subscribers.wardsintelligence.com/analysis/world-vehicle-population-rose-46-2016

[3] "Informe sobre la situación mundial de la seguridad vial 2015" Organización Mundial de la Salud, Francia, Informe Técnico: WHO/NMH/NVI/15.6, octubre 2015

[4] Morgan Stanley, "Nikola's Revenge: TSLA's New Path of Disruption", 25-02-2014

[5] C. Punzón, "Los coruñeses pierden 68 horas al año en atascos de tráfico, y 56 vigueses" *La Voz de Galicia*, 09-05-2018. **[En línea]. Disponible en:** https://www.lavozdegalicia.es/noticia/galicia/2018/05/09/coruneses-pierden-68-horas-ano-atascos-trafico-56-vigueses/0003_201805G9P4991.htm

[6] J. Campderrós-i-Canas, "¿Cuánto cuesta estar parado en el tráfico?", *Roshfrans*, noviembre 2017. **[En línea]. Disponible en:** http://www.roshfrans.com/cuanto-cuesta-estar-parado-en-el-trafico/

[7] P. Tejeiro, "Los coches autónomos podrían poner fin a los atascos", *Blogthinkbig*, 10-05-2017. **[En línea]. Disponible en:** https://blogthinkbig.com/los-coches-autonomos-podrian-poner-fin-los-atascos

[8] A. Ruiz García, "Sistemas de Percepción y visión por Computador", 02-02-2015

[9] Ibáñez, "Sistemas de detección en los coches para evitar accidentes", *Xataka*, 25-07-2011. **[En línea]. Disponible en:** https://www.xataka.com/automovil/sistemas-de-deteccion-en-los-coches-para-evitar-accidentes

[10] Anónimo, "Everything you need to know about Lane Departure Alert", *Toyota*, 07-03-2017. **[En línea]. Disponible en:** https://www.toyota.ca/toyota/en/connect/412/everything-you-need-to-know-about-lane-departure-alert

[11] Anónimo, "Bosch desarrolla una cámara estéreo para el frenado de emergencia", *Europa Press*, 04-05-2015, **[En línea]. Disponible en:** http://www.europapress.es/motor/coches-00640/noticia-bosch-desarrolla-camara-estereo-frenado-emergencia-20150504183526.html

[12] M. DeBord, "Automakers are making big investments in a key piece of self-driving-car technology that Tesla is avoiding", *Business insider*, 30-10-2017 [**En línea]. Disponible en:** https://www.businessinsider.com/ford-argoai-gm-invest-lidar-tesla-2017-10?IR=T

[13] S.Achatz, "State of the Art of Object Recognition Techniques", Facultad de electrónica e informática, Technische Universitat Munchen, 2016, **[En línea]. Disponible en:** https://www.nst.ei.tum.de/publikationen/nstseminarberichte/

[14] M.C. Munteanu, A. Caliman, C. Zaharia, "*Convolutional neural network*", 29-01-2016.

[15] M.A. Nielsen, "*Neural Networks and Deep Learning*", 2015.

[16] P. Velickovic, "Deep learning for complete beginners: convolutional neural networks with keras", *Cambridge Spark*, 20-03-2017, **[En línea]. Disponible en:** https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

[17] S.Ren, K. He, R. Girshick, J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Microsoft Research, 2015.

[18] J. Beltrán, et al., "BirdNet: a 3D Object Detection Framework from LiDAR information, *Universidad Carlos III*, Leganés, Madrid, 03-05-2018.

[19] C. R. Qi, W. Liu, C. Wu, H. Su, L. J. Guibas, "Frostum PointNets for 3D Object Detection from RGB-D Darta, IEEE Xplore, 2018.

[20] S.Scharstein et al., "High-resolution stereos datasets with subpixel-accurate ground truth", *German conference on Pattern Recognition (GCPR 2014),* Münster, Alemania, 09-2014.

[21] S.Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* New York, 06-2006.

[22] M. Cordts, et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding" *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] O. Russakovsky, et al., "ImageNet Large Scale Visual Recognition Challenge", *Springer US*, 11-04-2015.

[24] T. Lin, et al., "Microsoft COCO: Common Objects in Context", *Cornell University*, 21-02-2015.

[25] M. Menze, A. Geiger, "Object Scene Flow for Autonomous Vehicles", *Karlsruhe Institute of Technology*, Karlsruhe, 2015.

[26] Página web de Qt: https://www.qt.io/ [Último acceso: Agosto 2018]

[27] Página web de GNU: https://www.gnu.org/licenses/lgpl-3.0.en.html [Último acceso: agosto 2018]

[28] Página web de PCL: http://pointclouds.org/ [Último acceso: Agosto 2018]

[29] Página web de Open Source Initiative: https://opensource.org/licenses/BSD-3-Clause [Último acceso: agosto 2018]

[30] K. Ellis, et al. "Identifying Active Travel Behaviors in Challenging Environments Using GPS, Accelerometers, and Machine Learning Algorithms", *Frontier in Public Health*, abril-2014. **[En línea]. Disponible en:** https://www.researchgate.net/publication/262055313_Identifying_Active_Travel_Behaviors_in_Challenging_Environments_Using_GPS_Accelerometers_and_Machine_Learning_Algorithms

[31] Tabla de amortización simplificada. Agencia Tributaria. http://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresarios_individuales_y_profesionales/Rendimientos_de_actividades_economicas_en_el_IRPF/Regimenes_para_determinar_el_rendimiento_de_las_actividades_economicas/Estimacion_Directa_Simplificada.shtml [Último acceso: agosto 2018]

[32] A. Díaz Lucas, "Encuesta de Salarios y Actividad profesional", *Colegios oficiales de ingenieros industriales de Álava, Bizkaia, Gipuzkoa y Navarra*, 20-04-2017.

[33] Dirección General de Tráfico, "Instrucción 15/V-113", **[En línea]. Disponible en:** http://www.dgt.es/Galerias/seguridad-vial/normativa-legislacion/otras-normas/modificaciones/15.V-113-Vehiculos-Conduccion-automatizada.pdf